

Ein Markt & Technik Buch

Band 5:

Ein Leitfaden durch Simon's BASIC

Das Commodore 64 Buch

Hans Lorenz Schneider · Werner Eberl

Mit kommentiertem Listing

Das Commodore 64-Buch
Band 5

Hans Lorenz Schneider
Werner Eberl

Das Commodore 64-Buch

Band 5: Ein Leitfaden durch Simon's BASIC

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Schneider, Hans Lorenz:
[Das Commodore-vierundsechszig-Buch]
Das Commodore-64-Buch / Hans Lorenz Schneider;
Werner Eberl. — Haar bei München: Markt-und-Technik-Verlag
(Computer persönlich)
NE: Eberl, Werner:
ISBN 3-922120-71-7
Bd. 5. Ein Leitfaden durch Simon's BASIC. — 1984

15 14 13 12 11 10 11 9 8 7 6 5 4 3 2 1

88 87 86 85 84

ISBN 3-922120-71-7

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Vorwort

Immer wieder wurden wir gefragt: "Wann kommt denn endlich Band 5". Allein schon vom Umfang des Buches her werden Sie feststellen, daß wir Ihnen einiges bieten. Besonders das kommentierte Listing von Simon's Basic hat uns lange Zeit in Anspruch genommen. Über 16000 Byte von fremden Programmierern auseinanderzupflücken und festzustellen, was wo gemacht wurde, ist kein leichtes Unterfangen. Wir hoffen, Sie sind mit dem Ergebnis zufrieden.

Dieses Buch soll allen, die sich mit Simon's Basic beschäftigen, eine Hilfe sein. Sowohl für den Anfänger, der die Basic-Unterstützung nur zur Vereinfachung bei der eigenen Programmerstellung heranzieht, wie auch für den Profi, der einige Unstimmigkeiten in den Maschinenprogrammen beseitigen möchte.

Entgegen dem Handbuch haben wir eine andere Aufteilung gewählt. Zunächst werden alle Befehle noch einmal kurz besprochen, wobei wir auf Besonderheiten, die nicht im Handbuch stehen hinweisen. Auch werden hier die Einsatzmöglichkeiten der Befehle besprochen. In einem weiteren Teil werden die wohl am häufigsten gebrauchten Befehle aus Simon's Basic an ausführlichen Beispielen näher erläutert (Grafik, Sprites, Musik). Den Abschluß bildet das kommentierte Assembler-Listing. Hingewiesen sei auch auf die Befehlsübersichten im Anhang, die Ihnen sicherlich gut als Nachschlagewerk dienen.

Auch in diesem Buch möchten wir Sie wieder zu konstruktiver Kritik aufrufen.

An dieser Stelle sei auch allen Mitarbeitern der Schneider Software GmbH gedankt, die uns bei der Erstellung des Manuskriptes geholfen haben.

München, im Juli 1984

Hans Lorenz Schneider **Werner Eberl**

Einleitung

Wie bereits im Vorwort erwähnt, wollen wir uns mit diesem Buch an alle Qualifikationsschichten von Programmierern wenden, die mit Simon's Basic arbeiten. Die Kenntnis des Handbuches wird in diesem Buch vorausgesetzt, auch wenn wir an einigen Stellen Wiederholungen bringen, um Ihnen das Nachschlagen zu ersparen.

Im ersten Kapitel bringen wir eine Übersicht über alle Befehle von Simon's Basic, wobei wir auch auf Besonderheiten eingehen, die nicht im Handbuch stehen. Das Kapitel ist nach den Einsatzbereichen der Befehle gegliedert. Besonders hinweisen wollen wir auf Kapitel 1.11, wo wir noch einige Befehle aufführen, die nicht in jedem Handbuch stehen.

Das zweite Kapitel widmet sich voll und ganz den Grafikbefehlen, wobei wir zunächst auf normale Grafik-Befehle eingehen. Im weiteren besprechen wir dann die Zusammenarbeit von Texten und Grafikmodus sowie Sprites und Grafikmodus. Es folgen zwei Kapitel mit Balken- und Liniendiagrammen und ein Kapitel über Joystick und Paddles. Zum Abschluß noch ein Kapitel über die Definition von neuen Zeichensätzen.

Ganz den Sprites zugeordnet ist Kapitel drei, wobei hier das Beispiel aus Band 1 und Band 3 fortgesetzt wird. Die Gliederung entspricht dabei auch der Vorgehensweise bei der Erstellung eines Programmes mit Sprites.

Zwei kurze Kapitel beschäftigen sich mit den Fehlermeldungen - die nicht im Handbuch beschrieben sind - und der Musik, wobei wir hier auch auf dreistimmiges Spiel eingehen.

Den größten Umfang hat Kapitel sechs, wo wir das kommentierte Assembler-Listing abbilden. Zunächst dazu jedoch noch einige Vorbemerkungen und anschließend dazu die Tabelle der verwendeten Namen und Symbole sowie eine Übersicht der von Simon's Basic verwendeten Zero-Page-Adressen.

Im Anhang befinden sich zunächst zwei Listings aus Band 3, um Ihnen die Vergleichsmöglichkeiten bei den entprechenden Kapiteln zu geben. Es folgen eine Übersicht über die ver-

wendeten Parametertypen und die Bildschirm-Codes, damit Sie die Listings auch richtig interpretieren können. Sehr wichtig für Sie als Nachschlagewerk sind die beiden Übersichten in Anhang 5 und Anhang 6, die Ihnen einerseits alle Befehle mit ihren Parametern aufzeigen, andererseits die Fundorte im Handbuch und in dem vorliegenden Buch aufzeigen, sowie eine Kurzbeschreibung darstellen.

Der SECURE-Befehl kann durch ein kurzes Basic-Programm aufgehoben werde, das wir Ihnen in Anhang 7 vorstellen.

Das Commodore 64-Buch

Band 5: Ein Leitfaden durch Simon's Basic

Inhaltsverzeichnis

1

Die Befehle von Simon's Basic — eine Kurzvorstellung

1. Befehlsübersicht

Im ersten Kapitel wollen wir die zusätzlichen Befehle des Simon's-Basic in ihrer Gesamtheit erläutern und Unstimmigkeiten gegenüber dem Handbuch aufzeigen. Simon's Basic bietet sehr viele wichtige Befehle. Jedoch wäre es bei manchen Befehlen wünschenswert, wenn sie mehr Möglichkeiten zulassen würden. Z.B. ist ein RENUMBER ohne eine Veränderung der Sprungbefehle hinter GOTO und GOSUB kein ausreichendes Hilfsmittel.

Im Anhang ist eine Übersicht über alle Befehle mit einer Kurzbeschreibung ihrer Bedeutung dargestellt. Diese Übersicht kann auch als 'Handzettel' für diejenigen dienen, die schon mit Simon's Basic arbeiten. Weiterhin enthält die Liste die Fundstellen aller Befehle im Handbuch und dem vorliegenden Buch (Seitenangabe).

Simon's Basic enthält viele dringend notwendige Befehle, aber auch Befehle, die wohl nur in sehr seltenen Fällen benutzt werden. Auf jeden Fall ist Simon's Basic für den geübten Programmierer eine wertvolle Unterstützung. Besonders hervorzuheben sind hier die Befehle, die unter Programmierhilfen zusammengestellt sind. Weiterhin einige Befehle zur Verarbeitung von Zeichenreihen wie z.B. PLACE. Für Programmierer, die auch andere Programmiersprachen wie z.B. PL/I oder PASCAL kennen, dürften besonders die Strukturbefehle und die ERROR-Befehle interessant sein.

Um die speziellen Möglichkeiten des Commodore 64 wie die hochauflösende Grafik, die Definition von Sprites und den Sound-Generator zu benutzen, sind natürlich die entsprechenden Befehle notwendige Voraussetzung, wenn programmieren nicht in Byte-Fummelei ausarten soll und keine aneren Hilfsprogramme (z.B. in den Bänden 1, 3, 4 und 7) vorliegen.

Zu den Befehlen, die wohl nur dann angewendet werden, wenn ein Programmierer auch alle Register des Computers ziehen will, gehören neben einigen Befehlen aus den anderen Bereichen bestimmt auch alle Befehle der Bildschirmsteuerung. Alles in allem kann man jedoch sagen: der zusätzliche Befehlsvorrat von Simon's Basic läßt kaum Wünsche offen.

Für den genauen Ablauf der Befehle sei auf das kommentierte Listing verwiesen.

1.1 Programmierhilfen

AUTO

Dieser Befehl ist versierten Programmierern von anderen Kits bestimmt schon hinlänglich bekannt. Er ermöglicht die zeilenweise Programmeditierung, ohne jeweils eine neue Zeilennummer eintippen zu müssen. Dies erspart hauptsächlich beim fließenden Eintippen eines Programms die Überlegung: Welches ist denn jetzt die nächste Zeile?

COLD

Dieser Befehl ersetzt das Ein- und Ausschalten des Computers, wenn ein Kaltstart durchgeführt werden soll. Intern werden im Rechner immer Zeiger verwaltet, die auf den Anfang des Programms, den Anfang des variablen Speichers, usw. zeigen (vgl. Band 4). Der Befehl COLD bewirkt nichts anderes, als das Rücksetzen dieser Zeiger in den Ausgangszustand. Simon's Basic bleibt dabei eingeschaltet.

DELAY

Mit dem DELAY-Befehl kann die LIST-Geschwindigkeit eingestellt werden. Daß hier 256 Möglichkeiten zur Verfügung stehen, ist mehr als ein Programmierer benötigt. Prinzipiell wird sich jeder aus den Möglichkeiten ein oder zwei Geschwindigkeiten aussuchen, die seiner Lesegeschwindigkeit am Bildschirm entsprechen. Im Prinzip baut dieser Befehl in die Basic-List-Routine eine Warteschleife ein.

DISAPA

In Verbindung mit dem Befehl SECURE ist der Befehl DISAPA ein Mittel, um sein Programm gegen unbefugtes Auslisten zu schützen. Im Prinzip wäre es möglich das gesamte Programm mit diesem Befehl zu schützen, jedoch macht man sich selbst die Arbeit der Softwarepflege damit nur schwieriger. Sinnvoll wäre es, diesen Befehl in einem kurzen Programmstück zu verwenden, welches einige andere Sicherungsmethoden enthält.

Mit einem kurzen Basic-Programm (siehe Anhang) kann man jedoch den Schutz aufheben.

DISPLAY (bzw. KEY0)

Eine reine Informationsanweisung, die aber sehr wichtig ist, da es sonst sehr schwierig wäre, die aktuelle Belegung der Funktionstasten festzustellen.

DUMP

Der Vorteil eines Interpreters liegt zu einem großen Teil darin, daß während eines Programmlaufes das Programm abgebrochen werden kann, und die Variablen abgefragt werden können. Dies erleichtert das Austesten gegenüber Compilerversionen erheblich. Nun ist es recht mühsam, immer nach einem BREAK im Programm ein PRINT-Befehl für alle -oder auch nur die benötigten- Variablen einzugeben, wenn mehrere sogenannte Break-Points gesetzt sind. Diese Arbeit erleichtert der DUMP-Befehl. Allerdings werden Matrizen (Arrays) nicht angezeigt.

FIND

Ähnlich dem DUMP-Befehl erleichtert der FIND-Befehl das Testen sowie das Dokumentieren von Programmen. Besonders bei langen Listings ist es sehr mühsam, das gesamte Programm nach einer bestimmten Variablen zu durchsuchen. Da in Basic auch im Prinzip alle Variablen global sind, dürften - außer temporären - den Variablen nicht mehrfache Bedeutungen zugewiesen werden. Mit dem FIND-Befehl ist es auch möglich, zu prüfen, ob eine Variable schon im Programm vorhanden ist oder nicht. Leider wird nur die Zeilennummer und nicht die Zeile selbst angezeigt.

KEY

Da der Commodore Funktionstasten anbietet, ist es auch sinnvoll diese mit häufig verwendeten Basic-Befehlen (z.B. LIST) zu belegen. Das Löschen eines Key ist nur mit KEY(nr)," " (Leerzeichen) möglich. Ein Beispiel für die verkürzte Ausgabe auf Drucker ist nachfolgend aufgeführt (Ausgabe mit DISPLAY und HRDCPY). Dabei ist zur Ausgabe f1 und anschließend f3 zu drücken. Für die verwendeten Befehle wurden deren Abkürzungen benutzt. Mit POKE $C646,10 werden alle Key-Funktionen abgeschaltet.

```
KEY1,"O┐1,4:C╲1:L╮"+CHR$(13)
KEY2,""
KEY3,"P─1:CLOSE1"+CHR$(13)
KEY4,""
```

MERGE

Der MERGE-Befehl ermöglicht zwar das Einkopieren von anderen Programmen in ein Programm, das sich im Hauptspeicher befindet, jedoch läßt dieser Befehl einige Möglichkeiten vermissen. Z.B. ist das Laden von bestimmten Programmteilen eines Programms von Diskette nicht möglich. Dies ist besonders ein Nachteil, wenn aus anderen Programmen nur bestimmte Unterprogramme übernommen werden sollen. Außerdem muß die erste Zeilennummer des neuen Programmes eine höhere sein, als die letzte Zeilennnummer des ersten Programmes, was wohl selten der Fall ist.

OLD

Ab und zu kann es vorkommen, daß versehentlich ein NEW-Befehl eingegeben wurde, und man feststellt, daß das Programm vorher nicht abgespeichert bzw. die Kontroll-Lampe an der Floppy blinkt. Da durch den NEW-Befehl nur Zeiger intern umbesetzt werden, ist eigentlich noch nicht alles verloren. Aber es ist doch sehr mühsam das Programmende des Programms und die Werte für den Beginn der Variablentabelle usw. ausfindig zu machen. Dies erspart einem der OLD-Befehl, allerdings sind die Variablen trotzdem verloren.

OPTION

Eine Anwendungsmöglichkeit für diesen Befehl, der alle Befehle von Simon's-Basic hervorhebt, ist direkt nicht ersichtlich. Nützlich ist er vielleicht, wenn ein Programm in normales Basic umgeschrieben werden soll. Aber wenn jemand ein Programm, das mit Simon's Basic erstellt wurde, erhält, und dies umschreiben will, weil ihm die Programmierunterstützung nicht zur Verfügung steht, der könnte diesen Befehl gebrauchen. Aber wer gibt schon seine Programme weiter mit einer Liste: hier sind die Befehle, die geändert werden müssen.

PAGE

Da der Bildschirm des Commodore 64 nur 40 Zeichen je Zeile hat, und das Auslisten der Programme doch etwas schneller geht als bei den Modellen der alten Serien, verschwinden

Programmstücke nach oben aus dem Bildschirm heraus schneller, als man evetuell die STOP-Taste gefunden hat. Dies kann man einerseits mit Benutzung der CTRL-Taste beeinflussen, andererseits mit dem weiter oben beschriebenen DELAY-Befehl. Komfortabel ist es natürlich, wenn man vor Beginn einer jeden Programmiersitzung den Befehl PAGE verwendet, womit ein Blättern in Vorwärtsrichtung seitenweise erzielt werden kann.

RENUMBER

Wo fast jedes auf dem Commodore 64 erstelltes Programm dynamisch wächst, wird mal hier eine Zeile eingefügt, mal wird dort eine Zeile herausgenommen. Um dieses ganze Zeilennummernwirrwarr in den Griff zu bekommen ist natürlich der RENUMBER-Befehl sehr nützlich. In mühsame Kleinarbeit artet es jedoch aus, wenn Sie anschließend alle Sprungadressen bei GOTO/GOSUB-Befehl von Hand ändern müssen. Hier wäre eine erweiterte Version wünschenswert.

SECURE

Dieser Befehl bewirkt nur das eigentliche Schützen, der durch den Befehl DISAPA gekennzeichneten Befehle.

TRACE/RETRACE

Zum Testen von Programmen - besonders bei sogenannten Endlos-Schleifen - leistet der TRACE-Befehl, mit dem die aktuelle Zeilennummer eines laufenden Programmes angezeigt wird, sehr nützliche Hilfe. Der TRACE-Befehl funktioniert nicht nach MEM.

1.2 Strukturbefehle und ERROR-Befehle

Diese Befehle lassen sich nicht im einzelnen genügend erklären, so daß wir diese im Zusammenhang besprechen wollen. Für REPEAT, LOOP und EXEC existiert je ein Stack (Kellerspeicher), der bis zu fünf Ebenen erlaubt.

Schleifen - bedingte Schleifen - bedingte Anweisungen

Der erste Bereich der Strukturbefehle widmet sich den Schleifen und bedingten Anweisungen. Da das normale Basic nur IF...THEN-Befehle zuläßt ist es eine wesentliche Ver-

einfachung, wenn diese Befehle auch einen **ELSE-Teil** erhalten. Dadurch können aufwendige Konstruktionen mit GOTO-Befehlen vermieden werden, wie Bild 1.2.1 zeigt. Bild 1.2.2 zeigt die Bedingungen bei einem IF-Statement, die sehr komplex sein können, so daß es sinnvoll ist diese Bedingung in einem weiteren Befehl ohne erneute Eingabe wieder prüfen zu können. Dies kann man mit dem Befehl **RCOMP... ELSE**, da immer noch nicht, wie in anderen Programmiersprachen, eine blockweise Bearbeitung in verschiedenen Zeilen der THEN- /ELSE-Teile erfolgen kann.

Interne Funktionsweise: beim Simon's-IF-Befehl wird ein Flag gesetzt, das bei RCOMP nur abgefragt wird. ELSE muß in der gleichen Zeile stehen.

```
100 REM OHNE IF...THEN...ELSE
110 IF A=B THEN C=D : GOTO 130
120 E=F
130 REM FORTSETZUNG
140 :
150 :
160 :
170 REM MIT IF...THEN...ELSE
180 IF A=B THEN C=D ELSE E=F
190 REM FORTSETZUNG
```

Bild 1.2.1 : Programmierung für bedingte Anweisungen mit GOTO und mit IF...THEN...ELSE

```
100 REM OHNE RCOMP...ELSE
110 IF A=B AND X<Y OR F>G THEN PRINT"SEHR LANGER TEXT";
120 IF A=B AND X<Y OR F>G THEN PRINT"DER NICHT IN EINE";
130 IF A=B AND X<Y OR F>G THEN PRINT"ZEILE PASST. !!!!"
140 IF A=B AND X<Y OR F>G THEN GOTO 160
150 PRINT "NOCH EIN TEXT"
160 REM FORTSETZUNG
170 :
180 :
190 :
200 REM MIT RCOMP...ELSE
210 IF A=B AND X<Y OR F>G THEN PRINT"SEHR LANGER TEXT";
220 RCOMP PRINT"DER NICHT IN EINE ZEILE PASST.";
230 RCOMP PRINT" !!!!" : ELSE PRINT "NOCH EIN TEXT"
240 REM FORTSETZUNG
```

Bild 1.2.2 : Umfangreiche Bedingungen und ihre verkürzte Wiederholung

Eine weitere Verbesserung ist die Programmierung von Schleifen mit Bedingungsteilen. Das Beispiel im Handbuch ist relativ ungünstig gewählt, da dieses Beispiel durch eine einfache FOR...NEXT-Schleife ersetzt werden kann. Bild 1.2.3 zeigt einen sinnvollen Einsatz für den Befehl **REPEAT...UNTIL.**

```
100 REM VERGLEICH ZWEIER ZAHLEN ALS ABBRUCHKRITERIUM
110 REPEAT
120 ZN = ZN - ZA
130 UNTIL ABS(ZN-ZA) < 0.0000001
```

Bild **1.2.3 :** Beispiel für REPEAT...UNTIL

Dabei wird die Schleife abgebrochen, wenn eine Bedingung erfüllt ist, die nicht in einer FOR...NEXT-Schleife programmiert werden kann. Sicherlich ist es auch bei einfachen FOR...NEXT-Schleifen möglich, diese Schleifen mit einer IF-Abfrage zu verlassen, jedoch wird das Programm durch die neuen Befehle viel übersichtlicher. Ähnliches leistet auch der Befehl **LOOP...EXIT IF...END LOOP.**

Prozeduren

Sehr schön handhaben läßt sich die Verwendung von Unterprogrammen als Prozeduren mit Simon's Basic. Wie in blockorientierten Sprachen existiert auch ein Befehl **PROC**, der praktisch die Marke eines Unterprogrammes ist. Das Unterprogramm (in diesem Fall spricht man von Prozedur) wird auch nicht mit RETURN beendet, sondern mit **END PROC.** Der Aufruf kann sowohl mit **CALL** als auch mit **EXEC** erfolgen, wobei CALL einem GOTO entspricht (eine unübliche Art des Aufrufs einer Prozedur, da Prozeduren normal unabhängig von ihrer Lage im Programm ausgeführt werden) und EXEC einem GOSUB.

Wenn auch keine Blockvariablen im ursprünglichen Sinne zugelassen sind, kann man doch mit dem Befehl **LOCAL** Variableninhalte retten und später mit dem Befehl **GLOBAL** wieder auf diese Werte zurückgreifen. Dies erleichtert insbesondere die Programmierung großer komplexer Programme mit vielen Prozeduren. LOCAL darf nur einmal vor GLOBAL gegeben werden.

Fehlerbehandlung

Die Befehle **ON ERROR, ERRN, ERRLN und OUT** erlauben eine relativ komfortable Fehlerbehandlung. Die normale Fehler-

behandlung (Programmabbruch mit Anzeige des Fehlers) ist in den meisten Fällen nicht sehr benutzerfreundlich, da die Fehler per Programm abgefangen und durch eine entsprechende Benutzermitteilung eventuell auch behoben werden könnten. Mit dem Befehl ON ERROR ist eine solche komfortable Fehlerbehandlung in Abhängigkeit des aufgetretenen Fehlers (Liste im Handbuch) möglich. Achtung: **NO ERROR** schaltet nur ON ERROR ab, OUT gibt die Standard-Fehlermeldung aus.

1.3 Grafik-Befehle

Die Grafik-Befehle sollen nicht in ihrer alphabetischen Ordnung besprochen werden, sondern in ihrer natürlichen Anordnung, wie sie eventuell eingesetzt werden könnten. Die Grafik-Befehle sind für die Programmierung der hochauflösenden Grafik eine unabdingbare Voraussetzung. Dies heißt nicht, daß diese unbedingt dem Simon's Basic entnommen werden müssen, jedoch sollten diese oder ähnliche Befehle unbedingt zur Programmierung herangezogen werden. In den Bänden 1, 3 und 7 wurden bereits solche Erweiterungen besprochen.

Der Vorteil dieser Befehle liegt nicht nur in der einfacheren Programmierung, sondern - und dieser Vorteil ist nicht unerheblich - auch in der schnelleren Bearbeitung. Bei komplizierten Grafiken, wo viele Punkte gesetzt werden müssen, wartet man nicht selten mehr als eine Stunde, bis diese Grafik erzeugt ist, wenn man normales Basic heranzieht. Durch die Grafik-Befehle ergibt sich hier eine zigfach höhere Geschwindigkeit.

HIRES

Mit diesem Befehl wird im Simon's Basic die Grafik eingeschaltet und der Grafikspeicher gelöscht. Gleichzeitig werden für einfarbige Grafiken die Hintergrund- und Punktfarbe festgelegt. Auf die verschiedenen Speicher des Commodore 64 für die Farbgrafikdarstellung soll später eingegangen werden. Näheres zur Grafikbearbeitung finden Sie in den Bänden 1, 3 und 7.

MULTI

Da der Befehl HIRES nur einfarbige Grafiken zuläßt, kann man mit dem Befehl MULTI drei Farben zur gleichzeitigen Darstellung auf dem Bildschirm bestimmen. Der Video-Con-

troller ist auf MCM (Multi-Color-Modus) umgeschaltet und die zulässigen X-Werte sind auf 159 begrenzt. Vergleiche auch Band 3 dieser Buchreihe.

LOW COL

Mit diesem Befehl sind noch weitere drei Farben für den Multi-Color-Modus zuschaltbar. Während des Programms können zwar jedoch immer nur drei Farben angesprochen werden, es erscheinen mit weiteren LOW COL-Befehlen **alle** Farben auf dem Bildschirm. LOW COL funktioniert auch im Einfarben-Modus; natürlich werden nur die ersten beiden Farben (Vordergrund und Hintergrund) umgestellt.

HI COL

Mit dem Befehl HI COL wird der LOW COL-Befehl rückgängig gemacht. Für sechs - oder mehr - Farben sind also die Befehle LOW COL und HI COL entsprechend im Programm zu setzen.

PLOT

Der Befehl PLOT dient zur Ausgabe eines einzigen Punktes auf dem Bildschirm. Daß dies kein einfaches Unterfangen ist, haben wir auch in Band 1 schon dargestellt. Hierzu trägt der Aufbau des Grafikspeichers erheblich bei.

Die Statusvariable ST wird auf 8 gesetzt, wenn der zulässige Bereich für Koordinaten überschritten wurde. Ist LOW COL aktiv, wird derjenige Zeichenbereich umgefärbt, der den Punkt enthält.

LINE

Der Befehl LINE zeichnet bei seiner Verwendung eine Linie mit den angegebenen Parametern auf den Bildschirm.

REC, CIRCLE, ARC, ANGL, BLOCK

Ähnlich dem Ziehen einer Linie werden durch diese Befehle ein Rechteck gezeichnet (REC), eine Ellipse (der Kreis ist ein Sonderfall der Ellipse / CIRCLE), Segmente (ARC), Radien (ANGL) und ausgefüllte Rechtecke (BLOCK) gezeichnet.

PAINT

Mit dem Befehl PAINT läßt sich eine vorgegebene Figur (Rahmen) mit einer Farbe ausfüllen. Bei entsprechend komplizierten Figuren (Konvex und konkav gebogene Randstücke, Aussparungen in der Mitte) ist das schnelle Ausfüllen eines vorgegebenen Rahmens im Basic auch ein topologisches Problem, was selbst einem einigermaßen geübten Programmierer auf Anhieb nicht gelingen wird.

ROT, DRAW

Mit dem Befehl DRAW kann eine Figur aus Linien zusammengesetzt werden, und mit dem Befehl ROT kann eine derartig gezeichnete Figur in verschiedenen Winkeln gedreht werden. Achtung: ROT muß immer vor DRAW gegeben werden.

CSET

Dieser Befehl hat gleich drei Funktionen:

- Zeichensatz umschalten (von Grafik auf Groß/Kleinschrift bzw. umgekehrt); HIRES ausschalten
- Grafik zurückholen
- Eine mehrfarbige Grafik zurückholen und dabei diese Grafik mit anderen Farben versehen (in Verbin dung mit MULTI).

CHAR, TEXT

Diese beiden Befehle sind besonders wichtig, da im Modus der hochauflösenden Grafik keine Buchstaben, Zahlen und sonstige Zeichen ausgegeben werden können. CHAR und TEXT Befehle ermöglichen einerseits das Ausgeben einzelner Zeichen (CHAR) sowie ganzer Textzeilen (TEXT), wobei die Position, die vertikale Größe und die Farbe der Zeichen in dem Befehl selbst angewählt werden können.

TEST

der TEST-Befehl gibt an, welche Farbe (0,1,2,3) der angegebene Punkt besitzt.

1.4 Sprite-Befehle

Auch die Sprite-Befehle wollen wir in der Reihenfolge

ihrer Verwendung besprechen.

DESIGN

Mit DESIGN wird ein - nach einem Klammeraffen - definiertes Bild in ein Bit-Muster umgesetzt. Dies gilt für Sprites (Typ 0 und 1) und für neue Zeichen (Typ 2 und 3).

(Klammeraffe)

Wenn man den Speicher für ein Sprite zugeteilt hat, muß dieses als nächstes definiert werden. Dies kann man durch 21 Zeilen, denen ein Klammeraffe vorangestellt ist und die jeweils eine Zeile des Sprites (12 oder 24 nebeneinanderliegende Punkte) enthalten.

CMOB

Dieser Befehl dient zum Definieren der Farben von Multi-Color-Sprites.

MOB SET

Mit dem Befehl MOB SET werden die Eigenschaften eines Sprites festgelegt. In Simon's-Basic bedeutet die Abkürzung MOB 'Movable Object Block'. Bei diesen Eigenschaften sind u.a. enthalten, die Farbe des Sprites, die Priorität gegenüber dem Hintergrund und ob hochauflösende Grafik oder Multi-Color-Grafik gewünscht wird.

MMOB, RLOCMOB

Die Befehle MMOB und RLOCMOB dienen zur Darstellung des Sprites bzw. zur Bewegung des Sprites. Dazu können Start- und Zielposition des Sprites angegeben werden, sowie die Geschwindigkeit, mit der sich das SPRITE über den Bildschirm bewegen soll.

MOB OFF

Hiermit wird ein Sprite wieder ausgeschaltet.

DETECT, CHECK

Diese beiden Befehle dienen zur Vorbereitung einer Kollisionsprüfung und der Kollisionsprüfung selbst. Auch diese beiden Befehle sind in Basic durch einen einfachen PEEK-Befehl sehr schnell zu realisieren, sie ersparen einem aber die umständliche Suche, welches Byte im Video-Controller-Chip die entsprechende Aufgabe wahrnimmt, und das Auseinanderziehen der einzelnen Bits.

Der CHECK-Befehl ist auch als Anweisung wie DETECT einsetzbar. Der Parameter hinter DETECT (bzw. CHECK als Statement) bestimmt die Art der Abfrage (Sprite/Sprite oder Sprite/Hintergrund). Dementsprechend hat die dem DETECT-Befehl folgende Abfrage mit CHECK zwei oder einen Parameter, welche die Nummern der Sprite, die untersucht werden sollen, ergeben. Das Ergebnis 0 bedeutet Kollision, 1 keine Kollision.

1.5 Musik-Befehle

Die Anwendung der Musik-Befehle und ihrer Eigenschaften werden am besten deutlich, wenn man sich das entsprechende Kapitel in diesem Buch ansieht. Aber auch hier einiges, was nicht im Handbuch steht.

PLAY

Der PLAY-Befehl wird über Interrupt ausgeführt. PLAY 1 beinhaltet eine Warteschleife im Vordergrundprogramm.

MUSIC

Mögliche Elemente:

Funktionstasten: Notendauer

Angaben hinter 'SHIFT+CLR' (Herz in Revers am Bildschirm):

'SHIFT+CLR'1	:	wählt erste Stimme aus
'SHIFT+CLR'2	:	wählt zweite Stimme aus
'SHIFT+CLR'3	:	wählt dritte Stimme aus
'SHIFT+CLR'G	:	löscht Key-Bit der aktiven Stimme
'SHIFT+CLR'T	:	SYNC-Bit enthält den entgegengesetzten Wert
'SHIFT+CLR'C	:	setzt alle Bits der Wellenform auf 0
'SHIFT+CLR'R	:	wiederholt das Stück; anhalten mit STOP oder PLAY 0
'SHIFT+CLR'		mit irgend einem anderen Zeichen wird ignoriert

Angesprochen werden können die 12 'normalen' Noten: c, C, d, D, e, f, F, g, G, a, A, und b, wobei die großen Buchstaben (mit SHIFT) halbe Noten (durch # erzeugt, cis, dis, ...) darstellen. Zusätzlich sind 7 Sondernoten mit der Commodore-Taste ansprechbar (in der Notensprache mit b auf der Notenlinie: des, es, ...): C=D, C=E, C=F, C=G, C=A, C=B, C=C. Jedes andere Zeichen ergibt eine Pause. Hinter dem Notenzeichen muß ein Wert für die Oktavangabe angegeben werden.

1.6 Befehle für Zeichenreihen

Von den Befehlen für Zeichenreihen wollen wir hier nur zwei Befehle besonders hervorheben:

PLACE

Der Befehl PLACE wird immer da verwendet, wo eine kleinere Zeichenreihe in einer größeren gesucht wird. Dieser wichtige Befehl fehlte bisher bei allen Commodore Basic-Generationen. Sicherlich hat jeder Programmierer, der regelmäßig Programme schreibt, sich für diesen Befehl schon ein eigenes Unterprogramm angefertigt.

USE

Mit dem Befehl USE wird den Programierern ein noch größerer Gefallen getan als mit dem Befehl PLACE. Wohl in fast jeden Programm müssen Zahlen formatiert ausgegeben werden. Bisher mußte dazu immer mühsam ein Basic-Unterprogramm geschrieben werden, das eine Zahl als Zeichenreihe behandelt und in eine kaufmännische Zahlendarstellung umwandelt. Was bei anderen Basic-Dialekten selbstverständlich ist, wurde für den Commodore-Rechner in Simon's-Basic realisiert.

1.7 Befehle für Zahlen

Die sechs auf Zahlen anwendbaren Befehle sind wohl für jeden Programmierer im mathematisch-technisch-wissenschaftlichem Bereich unentbehrlich. Dabei ist der Befehl **EXOR** kein eigentlicher Zahlbefehl. Er bildet im Prinzip nur eine weitere logische Verknüpfung neben den schon vorhandenen AND, OR und NOT. Trick-Programmierer benutzen diesen Befehl um zwei Zahlen bzw. Bit-Muster zu vertauschen, ohne Zuhilfenahme einer dritten Variablen, indem

sie diesen Befehl dreimal auf ein Zahlenpaar bzw. Bit-Musterpaar anwenden.

Die Befehle **MOD**, **DIV** und **FRAC** bilden die im mathematischen Gebrauch definierten Funktionen Modulo, Division ohne Rest und extrahieren von Nachkommastellen einer Dezimalzahl nach.

Gerade bei den mannigfaltigen Adressen im Video-Controller des Commodore 64 wo einzelne Bits in Registern gesetzt oder gelöscht werden müssen, um bestimmte Tätigkeiten zu erreichen oder zu unterlassen ist eine Funktion die binäre Darstellungen in Dezimalzahlen umwandelt, wie der Befehl '%' in Simon's-Basic unerläßlich. In ähnlicher Weise ist es der Befehl '$' für den Assembler-Programmierer.

1.8 Bildschirmsteuerung

Wie zu Beginn bereits erwähnt, werden die Befehle zur Bildschirmsteuerung wahrscheinlich nur von Programmierern verwendet werden, die auch das letzte aus ihrem Rechner herausholen wollen. Simon's-Basic bietet dazu sehr viele Möglichkeiten, die Bildschirmausgabe interessant zu gestalten.

FLASH, OFF, BFLASH, BFLASH 0

Mit diesen vier Befehlen läßt sich das Blinken von Bildschirmfarben und des Rahmens durchführen. Dazu wird allerdings vorher notwendig sein, bei der Bildschirmausgabe entsprechend die Farben von Untergrund abzuheben, die später blinken sollen. Der FLASH-Befehl verlangsamt die eingebaute Uhr und das Basic-Programm wesentlich. FLASH ist auch ohne Geschwindigkeitsangabe möglich.

FCHR, FCOL, FILL, INV

Mit diesen vier Befehlen können bestimmte Bildschirmbereiche mit Zeichen und/oder Farben gefüllt bzw. invertiert werden.

MOVE

Mit dem MOVE-Befehl können Bildschirmbereiche dupliziert werden. Bei geschickter Bildschirmausgabe durch Cursorsteuerung läßt sich auf diese Art und Weise auch eine

fensterweise Ausgabe wie bei den Rechnern der 8000er Serie erreichen. In diesem Falle können sogar die Fenster in einen anderen Bereich kopiert werden. Viertelt man z.B. den Bildschirm, so kann man Viertel für die normale Bildschirmausgabe verwenden, und der Anwender kann sich bei Bedarf diese Bildschirmausgabe in ein anderes Viertel kopieren um eventuell Datenvergleiche durchzuführen.

LEFTB, RIGHTB, UPB, DOWNB, LEFTW, RIGHTW, UPW, DOWNW

Simon's-Basic erlaubt weiterhin Bildschirmbereiche zu rollen. Dies kann analog zu den oben angeführten Befehlen nach rechts, links oder nach oben und unten geschehen. Sehr interessant ist z.B., daß auch Bereiche gerollt werden können, so daß der Bildschirm teilweise erhalten bleibt und in anderen Bereichen des Bildschirms fortlaufend Daten angezeigt werden können. Das Bildschirmrollen kann sowohl zyklisch als auch mit Nachziehen von Leerzeilen bzw.-spalten erfolgen.

SCRSV, SCRLD

Bildschirminhalte im normalen Modus können mit diesen beiden Befehlen auf Diskette gespeichert bzw. wieder geladen werden. Dies kann z.B. interessant sein, wenn auf dem Bildschirm Balkengrafiken dargestellt wurden, deren Daten erst mühsam errechnet werden mußten. Zur nochmaligen Anzeige eines solchen Diagramms braucht dann keine neue Berechnung durchgeführt sondern lediglich der Bildschirminhalt von Diskette geladen werden. Auch Farben werden gespeichert. Achtung: Datei 1 wird geschlossen.

COPY, HRDCPY

Mit den beiden Copy-Befehlen sind sowohl die Bildschirmausgabe von hochauflösender Grafik (COPY) als auch eines Bildschirmes im Normalmodus (HRDCPY) auf einen Drucker möglich. Dies sind zwei relevante Befehle, besonders das Hardcopy der hochauflösenden Grafik, da dies eine relativ umständliche Druckerprogrammierung erfordern würde. Auch hier wird die Datei 1 geschlossen.

1.9 Befehle für LIGHT-PEN, JOYSTICK und PADDLE

Mit den vier Befehlen zur Abfrage des Status der obengenannten externen Hilfsgeräte lassen sich sicherlich sehr

einfach Spiele programmieren. Statt umständlichen PEEK-Abfragen im Programm können die Werte der externen Geräte durch diese Befehle sofort erfragt und somit auch gleich weiter verarbeitet werden. Beachten Sie bitte, daß nur ein Joystick und ein Paddle-Paar verwendbar ist. PENY ist nur sinnvoll nach PENX.

1.10 Sonstige Befehle

AT

ermöglicht die Positionierung des Cursors. Auch möglich ist: A$=AT(Spalte,Zeile)B$, wobei die Positionierung schon während der Zuweisung erfolgt.

DESIGN2 bzw. DESIGN3

Ähnlich den Sprites können auch die Zeichen des normalen Zeichensatzes neu definiert werden. Mit dem Befehl DESIGN wird zunächst festgelegt welches Zeichen erstellt werden soll, anschließend müssen acht Zeilen folgen, die die Form des Zeichen beschreiben (erstes Zeichen ein Klammeraffe). Mit DESIGN3 werden Multi-Colour-Zeichen erstellt. Der Multi-Colour-Modus wird dann durch POKE GRAPHICS+22, PEEK(GRAPHICS+22) OR 16 eingeschaltet.

DIR, DISK

Mit dem DISK-Befehl können Befehle an die Floppystation unmittelbar übergeben werden, ohne OPEN, CLOSE und die entsprechenden Fehlerkanäle anzugeben. Mit DIR kann das Inhaltsverzeichnis einer Diskette ausgegeben werden, wobei auch Jokerzeichen zugelassen sind, so daß ein sogenanntes Pattern-Matching möglich ist. Dies bedeutet , daß man sich Programme oder Dateien, die ganz bestimmte Buchstabenkombinationen enthalten, auszugsweise auslisten lassen kann. Achtung: Datei 1 wird geschlossen.

FETCH

Der FETCH-Befehl ermöglicht es, kontrollierte Eingaben im Programm zuzulassen, ohne daß aufwendige INPUT-Routinen mit dem GET-Befehl geschrieben werden müssen.

INKEY

Sehr wichtig ist auch der INKEY-Befehl, da er abfragt, ob eine Funktiontaste gedrückt ist. Dies ermöglicht eine schnelle Bearbeitung durch den Anwender, da nicht immer RETURN nach einer Eingabe gedrückt werden muß, bzw. eine umständliche Überprüfung über den GET-Befehl entfällt.

LIN

Im Gegensatz zu AT zeigt LIN nur die aktuelle Cursorzeile an.

MEM

Für Anwender, die ihren Zeichensatz selbst programmieren bzw. diverse Zeichen verändern wollen, ist der MEM-Befehl interessant, da er ohne komplizierte USR-Funktion den ROM-Bereich des Zeichensatzes in einen RAM-Bereich verlegt. MEM bietet außerdem die Möglichkeit einen zweiten Bildschirmspeicher anzulegen. Mit NRM kann der alte Bildschirm zurückgeholt werden.

PAUSE

Sicherlich jeder Programmierer hat in seinem Programm irgendwo eine 'leere' FOR...NEXT-Schleife, um eine Anzeige eine gewisse Zeit am Bildschirm aufrecht zu erhalten. Einerseits ist das ein Programmiertrick, der die Dokumentation des Programms nicht wesentlich erleichtert, andererseits ist in diesen Schleifen nur eine ungenaue Zeitangabe möglich. Mittels des PAUSE-Befehls kann man nun das Programm für eine genaue definierte Anzahl von Sekunden anhalten und sogar zusätzlich noch eine Meldung drucken. Während der Pause kann das Programm nicht mit der STOP-Taste abgebrochen werden. Einen PAUSE-Befehl, der dies ermöglicht, haben wir in Band 1 vorgestellt.

RESET

Der RESET-Befehl ist nicht zu verwechseln mit irgendwelchen RESET-Tasten und/oder -Schaltern. In Simon's-Basic dient er zum Setzen eines Zeigers auf eine beliebige DATA-Zeile. Dies wird dann benötigt, wenn in einem Programm mehrere verschiedene DATA-Blöcke vorkommen, die teilweise oder ganz neu eingelesen werden müssen. Dies vereinfacht der RESET-Befehl.

1.11 Befehle, die nicht im Handbuch stehen

BCKGNDS

Syntax: BCKGNDS f1,f2,f3,f4

Semantik: BCKGNDS legt die Hintergrundfarben fest und schaltet auf ECM (Extended-Color-Mode), dabei werden von jedem Zeichen zwei Bit vom ASCII-Code abgezweigt: es steht somit nicht mehr der gesamte Zeichensatz zur Verfügung. Die Aufteilung der Farben:

- f1: normale Hintergrundfarbe
- f2: Hintergrundfarbe der Zeichen mit SHIFT-Taste
- f3: Hintergrundfarbe der REVERS-Zeichen und des Cursors (nicht der Schriftfarbe)
- f4: Hintergrundfarbe für Zeichen mit SHIFT-Taste im REVERS-Mode

NRM macht BCKGNDS rückgängig.

COLOUR

Syntax: COLOUR rf,hf

rf: Rahmenfarbe
hf: Hintergrundfarbe

Semantik: COLOUR setzt Rahmen- und Hintergrundfarbe und erspart somit das lästige POKE 53280,rf:POKE53281,hf.

DISABLE

Syntax: DISABLE

Semantik: Setzt ON KEY-Anweisung außer Kraft.

GRAPHICS

Syntax: GRAPHICS

Semantik: Liefert Konstante $D000 = 53248; Adresse VIC

NRM

Syntax: NRM

Semantik: NRM macht MEM und BCKGNDS rückgängig.

ON KEY

Syntax: ON KEY Stringausdruck, diverse Anweisungen

Semantik: Wird eine Taste gedrückt, die im Stringausdruck des ON KEY-Befehls enthalten ist, so wird in den Anweisungsteil verzweigt. Die Tastatur wird dabei vor jedem Befehl abgefragt. Ein unbedingter Sprung erfolgt, wenn im Stringausdruck eine 'eckige Klammer zu' ($5D) enthalten ist.

RESUME

Syntax: RESUME

Semantik: RESUME funktioniert nur nach ON KEY. Bei RESUME wird das Programm beim ursprünglichen Befehl fortgesetzt. RESUME entspricht somit dem RETURN bei GOSUB.

SOUND

Syntax: SOUND

Semantik: Liefert Konstante $D400 = 53972; Adresse SID

2

Grafik-Befehle an Beispielen

2. Grafikbefehle an Beispielen

In diesem Kapitel wollen wir uns ausschließlich den Grafikbefehlen von Simon's Basic widmen. Zunächst wollen wir nur auf die eigentlichen Grafikbefehle eingehen und im zweiten Unterkapitel Text und Grafik mischen. Den Abschluß von Kapitel 2 bilden noch das umgeschriebene Programm für Balkendiagramme aus Band 3, um Ihnen eine Vergleichsmöglichkeit zwischen Simon's Basic und den in den anderen Büchern vorgestellten Grafikbefehlen zu geben, ein Programm für Liniendiagramme sowie einiges zum Thema Joystick und Paddles.

2.1 Beispiele mit Blöcken, Rechtecken und Kreisen

Im folgenden wollen wir einige kurze Beispielprogramme besprechen, die optisch ansprechende Grafiken auf den Bildschirm bringen. Neben dem Programm wird in der Regel auch ein Hardcopy abgebildet, das mit dem Befehl COPY von Simon's Basic erstellt wurde. Fangen wir mit einem einfachen Beispiel für Rechtecke an.

Rechteck

```
120 F=1
130 POKE53280,0
140 POKE53281,0
150 HIRES0,0
160 MULTI 1,7,8
200 FORI=0TO2*πSTEP.2
210 X1=30*SIN(I)
220 Y1=30*COS(I)↑7
230 X2=X1+30
240 Y2=Y1+30
250 REC   X1+50,Y1+80,X2,Y2,F
260 F=F+1
270 IFF=4THENF=1
280 NEXT
500 PAUSE5
998 PRINT"■
999 LIST
```

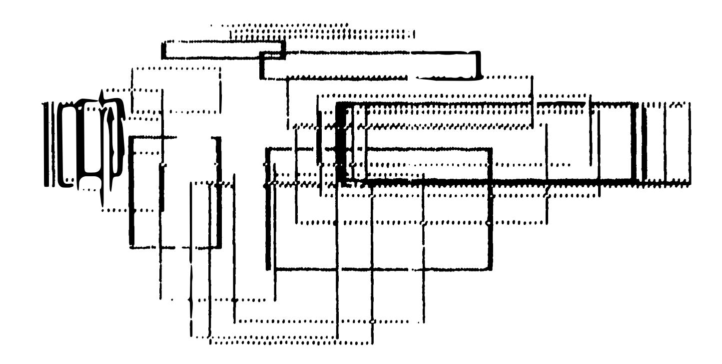


Bild 2.1-1 : Bildschirmcopy von RECHTECK

Zunächst wird die Variable F, die später für die Farbgebung im Grafikbefehl zuständig ist, initialisiert, dann werden Bildschirmhintergrundfarbe und Rahmenfarbe auf schwarz gesetzt. Darauf wird die Grafik mit dem HIRES-Befehl eingeschaltet, wobei hier Hintergrund und Zeichenfarbe gleich gewählt wurden, da anschließend mit dem Befehl MULTI die Farben (weiß, gelb und orange) festgelegt werden. Die Reihenfolge der Farbnummern ist wichtig, weil nachher in den Grafikbefehlen (über die Hilfsvariable F) die Ordnungszahl der Farbe im MULTI-Befehl angegeben wird.

Bevor wir jedoch im Programm weitergehen, betrachten Sie zunächst Bild 2.1-1. Hier noch etwas zu dem COPY-Befehl. Offensichtlich gibt dieser Befehl das im RAM abgelegte Bitmuster auf dem Drucker aus. Dadurch ist es z.B. auch möglich (wenn auch nur sehr schwer) dreifarbige Grafiken auf dem Drucker zu erkennen. Die Grafiken werden wie im internen Speicher auch, mit zwei ausgegebenen Punkten pro Bildpunkt verwertet. Bei den ersten beiden Farben kann man jedoch sehr schlecht unterscheiden, ob jeweils der rechte Punkt neben einem 'Leerpunkt' ausgegeben ist oder der linke Punkt.

Wenn Sie sich nun die Grafik anschauen, so werden Sie feststellen, daß die oberen linken Punkte der Rechtecke die Form einer nach innen eingedrückten Raute beschreiben. Dies kommt durch die Verwendung der mit sieben potenzierten Cosinus-Funktion. Wenn Sie das Bild weiter betrachten, werden Sie feststellen, daß von oben nach unten die Rechtecke immer höher und von links nach rechts immer breiter werden.

In der Schleife ab Zeile 200 bis Zeile 280 werden die

Daten für die Rechtecke errechnet. Zunächst werden in X1 und Y1 die Eckpunkte der oberen linke Ecke festgelegt und in X2 und Y2 anschließend die Seitenlängen des Rechteckes. In Simon's Basic ist die Ausgabe von Rechtecken mit der Ausgabe von Blöcken nicht direkt vergleichbar, da bei Blöcken jeweils die oberen linken und unteren rechten Ecken angegeben werden, bei Rechtecken aber die Seitenlängen.

Innerhalb der Schleife wird auch die Farbvariable jeweils um eine Farbe weiter gesetzt, bzw. auf die erste Farbe zurückgesetzt, wenn alle Farben durchlaufen wurden (Zeile 260 und 270).

Den Abschluß des Programms bildet ein Befehl, der die Programmausführung um fünf Sekunden anhält (Zeile 500), ein Befehl, der die Ausgabefarbe am Bildschirm auf weiß schaltet und zum Abschluß ein Befehl, daß das Programm wieder auslistet. Diese drei letzten Befehle dienen zum Aufrechterhalten der Grafik für fünf Sekunden (der Parameter kann von Ihnen beliebig geändert werden) und zur Ausgabe des Programmlistings falls Sie andere Programme, z.B. innerhalb der Schleife, ändern wollen. Diese Befehle sind an die meisten der folgenden Beispiele angehängt und sollen Ihnen das herumprobieren etwas erleichtern. Wir empfehlen das Ausprobieren verschiedener Parameter, damit Sie ein besseres Gefühl zwischen den doch recht abstrakten Parametern und der daraus resultierenden Grafik bekommen.

Block

```
1000 INPUT"VON                .■■■■";V
1010 INPUT"BIS                .■■■■";B
1020 INPUT"SCHRITTWEITE       .■■■■";S
1030 INPUT"HINTERGRUND        .■■■■";H
1040 INPUT"FARBE 1            .■■■■";F(1)
1050 INPUT"FARBE 2            .■■■■";F(2)
1060 INPUT"FARBE 3            .■■■■";F(3)
1070 POKE53281,H
1080 POKE53280,H
1090 HIRES7,7
1100 MULTI F(1),F(2),F(3)
1110 F=1
1120 FORI=VTOBSTEPS
1130 BLOCK I,I,2*I,2*I,1
1140 F=F+1
1150 IFF=4THENF=1
1160 BLOCK 150-2*I,I,150-I,2*I,3
1170 F=F+1
```

```
1180 IFF=4THENF=1
1190 BLOCK 150-2*I,100-I,150-I,100+I,2
1200 F=F+1
1210 IFF=4THENF=1
1220 BLOCK 150-2*I,200-2*I,150-I,200-I,1
1230 F=F+1
1240 IFF=4THENF=1
1250 BLOCK I,100-I,2*I,100+I,2
1260 F=F+1
1270 IFF=4THENF=1
1280 BLOCK I,200-2*I,2*I,200-I,3
1290 F=F+1
1300 IFF=4THENF=1
1310 NEXT
1320 PAUSE15
1330 LIST
```

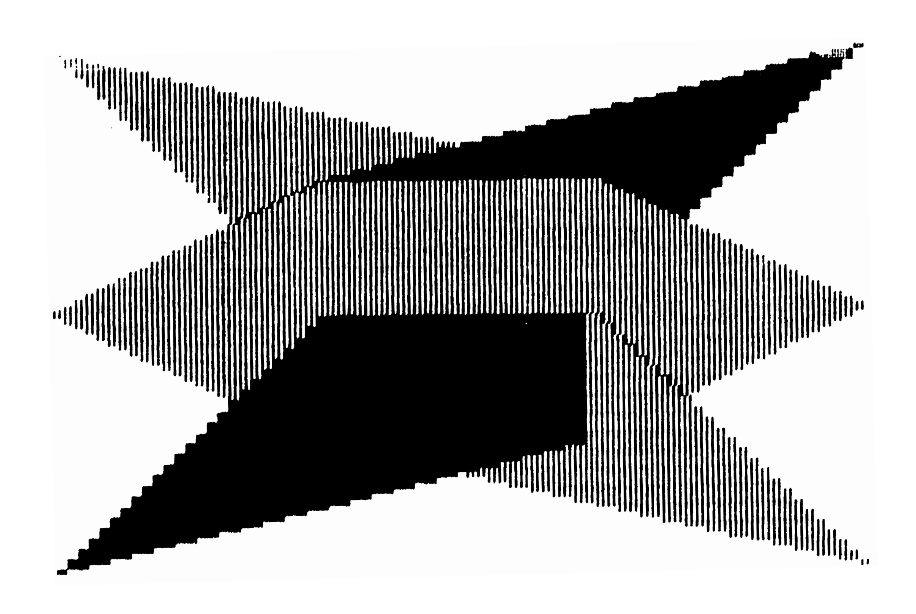


Bild 2.1-2 : Bildschirmcopy von BLOCK

Eine andere Möglichkeit, ein Programm mit verschiedenen Parametern öfter laufen zu lassen, wurde in diesem Beispiel gewählt. Hier werden die Parameter für die Ausgabe zunächst vom Bildschirm erfragt. Ob Sie nun jeweils Ihr Programm ändern wollen oder für jeden Testdurchlauf alle Variablen neu eingeben wollen, bleibt Ihnen überlassen.

Wie bereits erwähnt, wird in den Zeilen 1000 bis 1060 die nötige Information vom Bildschirm erfragt. Zunächt für die FOR...NEXT-Schleife die Start- und Endvariable sowie die Schrittweite, anschließend die Farben. Das Einschalten der Grafik und Umschalten auf hochauflösende Grafik geschieht genau wie im letzten Beispielprogramm. In diesem Beispiel hier wird deutlich, daß auch bei den Befehlen HIRES und MULTI Variablen statt konstanten Werten zwischen 0 und 15 auftreten dürfen.

Die Bestimmung der Farben läuft in diesem Programm ohne Auswirkungen mit, falls Sie die vorgegebenen Werte auch noch variabel halten wollen.

Aufgrund der Zahlwerte her müsste bei dem BLOCK-Befehl jeweils ein Quadrat erscheinen. Da in der hochauflösenden Grafik aber jeweils zwei benachbarte Punkte gleichzeitig angesprochen werden, ergibt sich durch die vorgegebene Programmierung ein Rechteck, was doppelt so breit wie hoch ist.

Die abgebildete Grafik wurde mit folgenden Parametern erstellt:

- Anfangswert für Schleife: 1
- Endwert für Schleife: 50
- Schrittweite der Schleife: 2
- Farben: 0, 6, 7, 8

Für die Ausgabe mit COPY ist natürlich die eingegebene Farbe nicht relavant.

Farbwechsel

```
1000 POKE53280,0
1010 POKE53281,0
1020 HIRES 0,0
1030 MULTI 6,7,7
1040 FORX=1TO75STEP3
1050 REC X,X,150-2*X,200-2*X,1
1060 REC X+1,X+1,148-2*X,198-2*X,2
1070 REC X+2,X+2,146-2*X,196-2*X,3
```

```
1080 NEXT
1090 PAUSE1
1100 FORI=50TO1STEP-1
1110 MULTI 7,7,6
1120 FORJ=1TOI:NEXT
1130 MULTI 7,6,7
1140 FORJ=1TOI:NEXT
1150 MULTI 6,7,7
1160 FORJ=1TOI:NEXT
1170 NEXT
1180 PAUSE10
1190 PRINT"█
1200 LIST
```

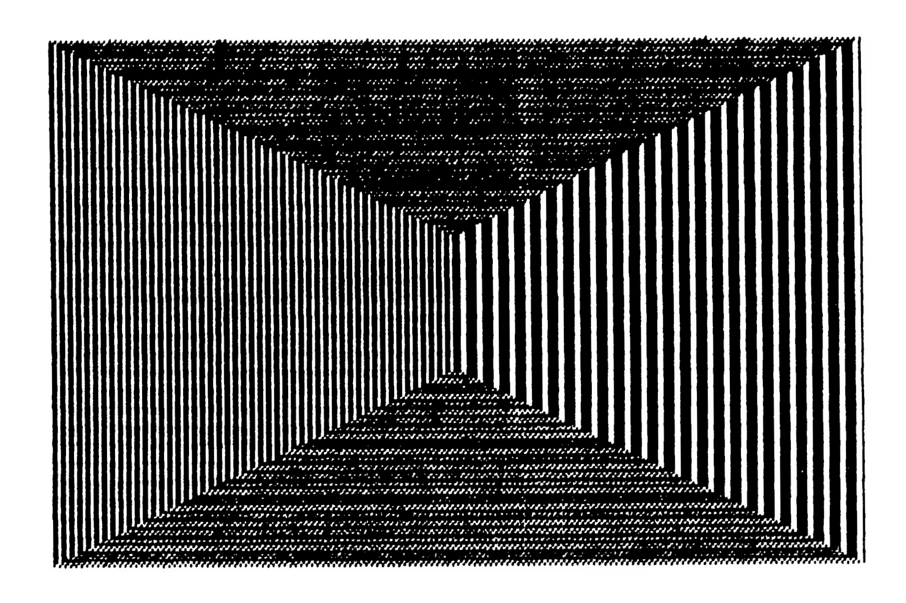


Bild 2.1-3 : Bildschirmcopy von FLUCHTREC

Im nächsten Beispielprogramm werden in einer Schleife jeweils drei Rechtecke ineinander gezeichnet, wovon zwei blau sind und eines gelb ist. Nachdem der Bildschirm gefüllt ist, werden über dem MULTI-Befehl jeweils Farbwechsel durchgeführt. Dieses Programm zeigt einerseits, daß zwei oder auch drei Farben im MULTI-Befehl gleich sein können. Obwohl z.B. bei den blaugezeichneten Rechtecken am Bildschirm kein Unterschied besteht, werden sie intern anders dargestellt. Diesen Umstand machen wir uns zu Nutze, indem wir in einer Schleife der ersten Farbe im

MULTI-Befehl die zweite Farbe zuordnen, der zweiten Farbe die dritte, und der dritten wieder die erste Farbe. In dieser Weise lassen wir in einer Schleife die Farben 'rotieren'. Etwas gestört durch die Bildschirmsynchronisation ergibt sich jedoch beim Durchlauf durch letztere Schleife der Anschein als würde man einen Gang entlanglaufen. Dies geschieht, indem man - optisch gesehen - zunächst dem Rechteck neben dem derzeitigen gelben auch Gelb zuordnet und das gelbe Rechteck dann blau einfärbt. Dadurch bekommen auch die beiden nebeneinanderliegenden blauen Rechtecke einen Sinn, da ja im Kreis getauscht wird.

Kreise, Ellipsen

Den gleichen Effekt haben wir auch im folgenden Programmlisting verwendet:

```
1000 POKE53280,0
1010 POKE53281,0
1020 HIRES 0,0
1030 MULTI 6,7,7
1040 FORX=75TO10STEP-6
1050 ARC 75,100,0,360,10,X,X,1
1060 ARC 75,100,0,360,11,X-1,X-1,2
1070 ARC 75,100,0,360,12,X-2,X-2,3
1080 NEXT
1090 PAUSE1
1100 FORI=100TO10STEP-1
1110 MULTI 7,7,6
1120 FORJ=1TOI:NEXT
1130 MULTI 7,6,7
1140 FORJ=1TOI:NEXT
1150 MULTI 6,7,7
1160 FORJ=1TOI:NEXT
1170 NEXT
1180 PAUSE10
1190 PRINT"♫
1200 LIST
```

Lassen Sie sich nicht durch die vielen Parameter beim ARC-Befehl abschrecken. Dem Befehl folgen zunächst die beiden Parameter für X-, und Y-Koordinate des imaginären Kreis-/Ellipsenmittelpunktes. Anschließend wird der Kreisausschnitt angegeben, der gezeichnet werden soll, gefolgt von der Genauigkeit der Auflösung. Geben Sie z.B. 90 als Schrittweite der Winkel an, so werden auf der Spitze stehende Quadrate/Rechtecke gezeichnet. Achtecke erhalten Sie mit Schrittweite 45. Je nach Größe der Figur dürften

Schrittweiten zwischen 5 und 10 für eine relativ runde Form durchaus ausreichend sein. Als weiteres folgen die beiden Parameter zur Bestimmung des Radius. Wählen Sie beide Radien gleich, so ergibt sich in normaler Grafik ein Kreis. Bei mehrfarbiger Grafik darf der X-Wert nur die Hälfte vom Y-Wert betragen, wenn Sie einen Kreis zeichnen wollen. Mathematisch gesehen ist der Kreis eine Sonderform der Ellipse. Wie bei jedem Grafikbefehl gibt der letzte Parameter die Farbe an.

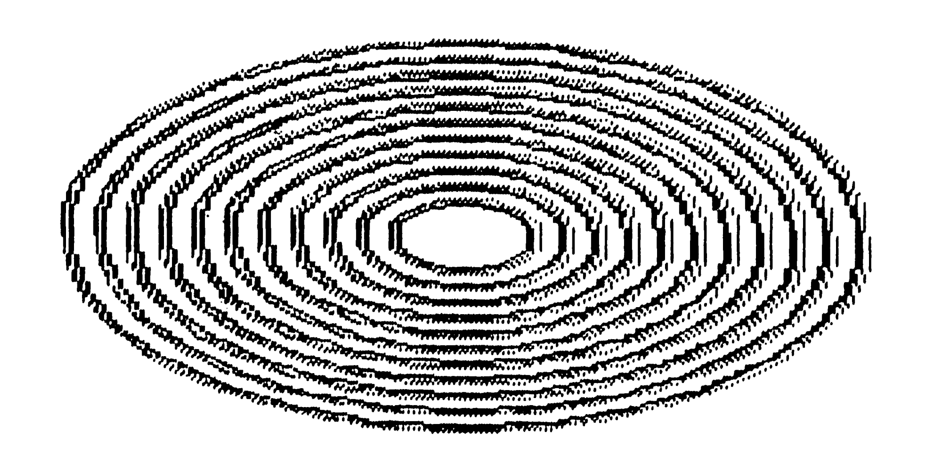


Bild 2.1-4 : Bildschirmcopy von FLUCHTELL

Die leeren FOR...NEXT-Schleifen in den Zeilen 1120, 1140 und 1160 können Sie auch durch einen PAUSE-befehl mit entsprechendem Parameter ersetzen. Durch die Abhängigkeit des Schleifenendekriteriums von der umgebenden Schleife und durch die Tatsache bedingt, daß die umgebende Schleife 'rückwärts' läuft, ergibt sich so eine immer schneller werdende Farbumsetzung, was optisch den Eindruck erweckt, daß die Form immer schnell auf einen zufließt.

Linie

Eine dritte Möglichkeit mit Grafikprogrammen zu probieren ist die Vorgabe von Variablen am Programmanfang, so daß man einzelne Variablen ändern kann, ohne jeweils alle Variablen über den Bildschirm eingeben zu müssen bzw. man an den einzelnen Programmstellen nicht immer alle Variablen austauschen braucht. Diese Form haben wir in den nächsten beiden Programmen realisiert.

```
1000 HIRES6,7
1010 SW=10
1020 V=80
1030 B1=120
1040 B2=140
1050 FORI=VTOB1STEPSW
1060 LINE 0,0,320,I,1
1070 NEXT
1080 FORI=VTOB2STEPSW
1090 LINE 0,0,I,200,1
1100 NEXT
1110 FORI=VTOB1STEPSW
1120 LINE 320,200,0,I,1
1130 NEXT
1140 FORI=VTOB2STEPSW
1150 LINE 320,200,I,0,1
1160 NEXT
1170 FORI=VTOB1STEPSW
1180 LINE 0,200,320,I,1
1190 NEXT
1200 FORI=VTOB2STEPSW
1210 LINE 0,200,I,0,1
1220 NEXT
1230 FORI=VTOB1STEPSW
1240 LINE 320,0,0,I,1
1250 NEXT
1260 FORI=VTOB2STEPSW
1270 LINE 320,0,I,200,1
1280 NEXT
1290 PAUSE5
1300 LIST
```

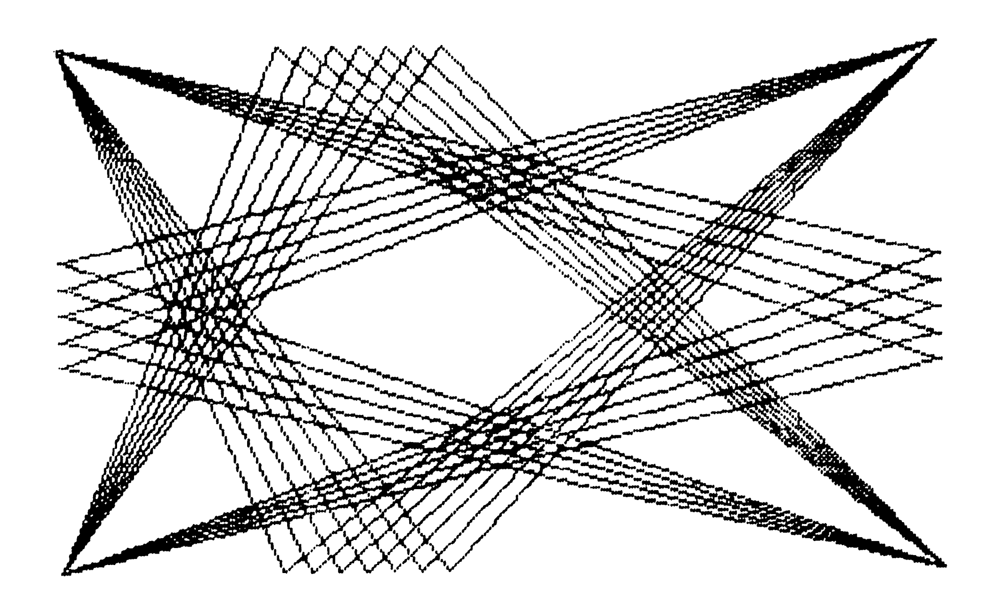


Bild 2.1-5 : Bildschirmcopy von LINE

Zunächst stellen wir aus Linien eine einfache Grafik dar, wie sie auch abgebildet ist. Im Prinzip werden immer von einem Eckpunkt weg einige Linien zu den gegenüberliegenden Rändern gezogen. Daß die Programme für den Mehrfarbenmodus nicht sehr stark geändert werden müssen, zeigt das folgende Beispiel, das ähnliches im Mehrfarbenmodus bewirkt:

```
1000 HIRES6,7
1010 MULTI 6,8,0
1020 SW=20
1030 V=60
1040 B1=140
1050 B2=160
1060 FORI=VTOB1STEPSW
1070 LINE 0,0,320,I,1
1080 NEXT
1090 FORI=VTOB2STEPSW
1100 LINE 0,0,I,200,1
1110 NEXT
1120 FORI=VTOB1STEPSW
1130 LINE 320,200,0,I,2
1140 NEXT
1150 FORI=VTOB2STEPSW
1160 LINE 320,200,I,0,2
1170 NEXT
1180 FORI=VTOB1STEPSW
1190 LINE 0,200,320,I,3
1200 NEXT
1210 FORI=VTOB2STEPSW
1220 LINE 0,200,I,0,3
1230 NEXT
1240 FORI=VTOB1STEPSW
1250 LINE 320,0,0,I,1
1260 NEXT
1270 FORI=VTOB2STEPSW
1280 LINE 320,0,I,200,1
1290 NEXT
1300 PAUSE5
1310 LIST
```

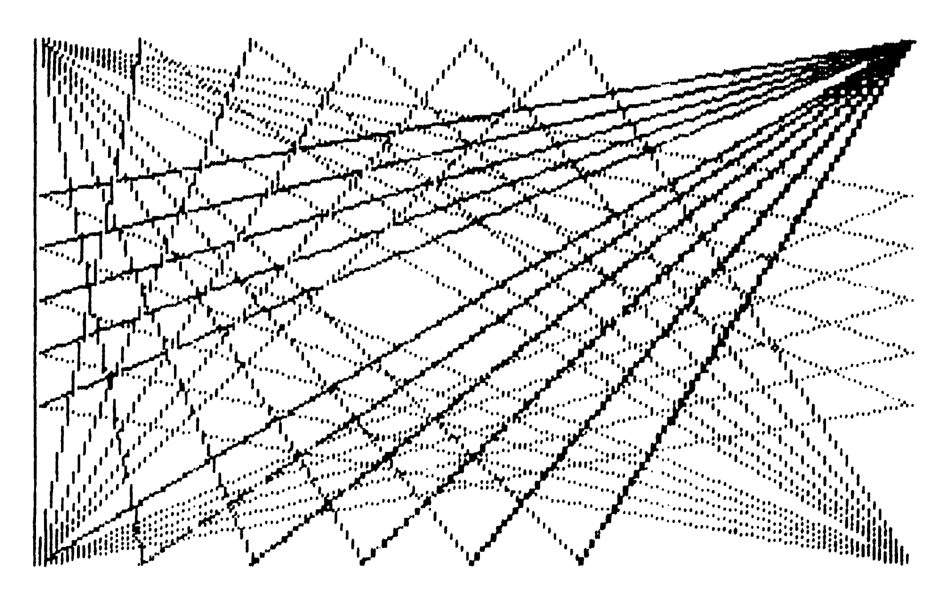


Bild 2.1-6 : Bildschirmcopy von LINEMULTI

Es wurden beide Grafiken abgebildet, um Ihnen die unterschiedlichen Auflösungen mit dem COPY-Befehl für Normal- und für Mehrfarbmodus aufzuzeigen.

DRAW, ROT

Die vorher beschriebenen Befehle BLOCK, REC und ARC sowie der Befehl CIRCLE lassen natürlich nur die grafische Ausgabe von fest vorgegebenen geometrischen Formen zu. Eigene Formen kann man mit der Befehlskombination ROT und DRAW erzeugen.

Wichtig: Der Befehl ROT muß auch gegeben werden, wenn weder Drehung noch Vergrößerung beabsichtigt ist, da sonst der Befehl DRAW nicht funktioniert.

Ob man sich seine Figuren aus kurzen Linien, kleinen Rechtecken und Blöcken oder ähnlichen Formen zusammensetzt oder mit DRAW, ist wohl Geschmackssache des Programmierers. Die Programmierung mit DRAW ist recht aufwendig. Zunächst müssen Zeichenreihen definiert werden, indem für jeden einzelnen Punkt gesagt werden muß, wo dieser relativ zum letzten Punkt stehen soll. D.h. Sie zeichnen ähnlich wie auf Papier, fortlaufend.

Nach einer kurzen Zeit hat man die für die verschiedenen Richtungen (und ob ein Punkt gesetzt werden soll oder nicht) benötigten Angaben (Ziffern 1 bis 8) auswendig im Kopf und man braucht nicht mehr nachschauen, was doch einiges an Zeit spart. Jedoch ist das Eintippen und Testen von mit dem DRAW-Befehl gezeichneten Formen recht mühsam. Wenn Sie den ROT-Befehl anwenden wollen, können Sie auch gerade Linien nicht mit dem LINE-Befehl ziehen, da diese nicht umgesetzt werden.

```
1000 HIRES6,7
1010 A$="66666666665555555555777777777788888888889"
1020 FORI=0TO7
1030 FORJ=1TO7
1040 ROT I,J
1050 DRAW A$,155,95,1
1060 PAUSE0.5
1070 NEXT
1080 NEXT
1090 PAUSE5
```

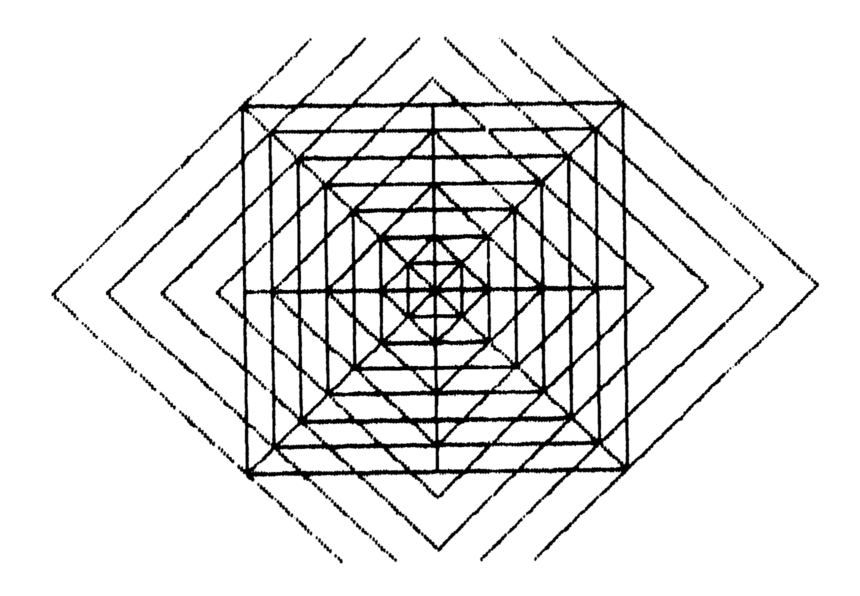


Bild 2.1-7 : Bildschirmcopy von DRAWROT

Wenn Sie ähnliche Passagen in Ihrer selbstgewählten Figur haben, so empfiehlt es sich, die von Basic zur Verfügung gestellten Zeichenreihenoperationen zu verwenden und Ihre Zeichenreihen so zu gestalten, daß sie teilweise wieder verwendet werden können.

Zu Demonstrationszwecken haben wir ein einfaches Quadrat herangezogen, das in Zeile 1010 definiert wird. In zwei ineinandergeschachtelten Schleifen wird anschließend jeweils die Figur gedreht und vergrößert.

Ausmalen von Formen

```
1000 HIRES 6,6
1010 MULTI 6,7,8
1020 A$="666666666655555555557777777777888888889"
1030 FORI=0TO7
1040 ROT I,4
1050 DRAW A$,80,100,1
1060 NEXT
1070 PAINT  80, 50,2
1080 PAINT  80,150,2
1090 PAINT  20,100,2
1100 PAINT 130,100,2
1110 PAINT   0,  0,1
1120 PAINT   0,199,1
1130 PAINT  75, 98,1
1140 PAINT  75,102,3
1150 PAINT  85, 98,3
1160 PAINT  85,102,1
1170 PAINT  78, 95,3
1180 PAINT  78,105,1
1190 PAINT  82, 95,1
1200 PAINT  82,105,3
1210 PAUSE10
1220 LIST
```

Zur Erstellung ansprechender Grafiken auf dem Bildschirm gibt es noch den PAINT-Befehl, mit dem begrenzte Flächen ausgemalt werden können. Achten Sie beim Ausmalen bitte darauf, daß die auszumalende Fläche keine 'undichte Stelle' aufweist. Der PAINT-Befehl erspart Ihnen mühsames Eintippen langer Zeichenreihen für den DRAW-Befehl. Mit dem DRAW-Befehl brauchen Sie so nur noch den Rahmen zu erstellen und diesen dann anschließend mit dem PAINT-Befehl auszufüllen.

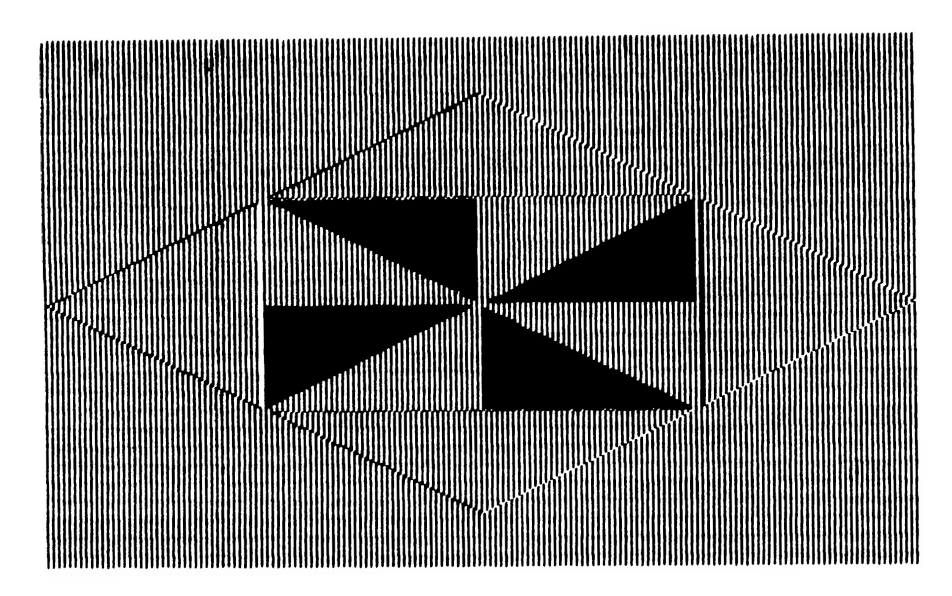


Bild 2.1-8 : Bildschirmcopy von PAINT

Durch Verwendung der Mehrfarbengrafik wurde aus unserem Quadrat im letzten Beispiel nun ein Rechteck bzw. eine Raute für die Drehung um 90 und 270 Grad. Mit den aufgeführten PAINT-Befehlen wird jeweils ein Punkt in den einzelnen Feldern lokalisiert, und das Unterprogramm in Simon's Basic füllt nachher bis zu den Grenzen diese Fläche aus.

In Bild 2.1-8 sieht man deutlich, daß die ersten beiden Farben im Multicolormodus bei einem Grafik-Hardcopy in der Regel nur an den Randlinien unterschieden werden können.

Wollen Sie zwischen Grafik und Textmodus innerhalb eines Programmes hin- und herschalten, ohne jeweils die Grafik neu zeichnen zu müssen - was besonders bei Erstellung einer Grafik mit Joystick recht mühsam ist, können Sie CSET 0 und CSET 1 verwenden. Ein Speichern von Grafiken wird von Simon's Basic nicht unterstützt. Eine mögliche vorgehensweise ist in Band 3 dargestellt.

2.2 Texte im Grafikmodus / Grafik und Sprites

In diesem Kapitel wollen wir einiges über das Mischen von Texten und Sprites mit dem Grafikmodus aufzeigen. Dazu haben wir als Beispiel ein Teil eines Roulette-Programms

gewählt, das das Spielfeld an einem Roulette-Automaten, wie Sie auch in Spiel-Casinos stehen, nachbildet. Hier werden nur mit den Zahlen 0 bis 12 gespielt, mit einer Anordnung, wie Sie in nachfolgendem Bild dargestellt ist.

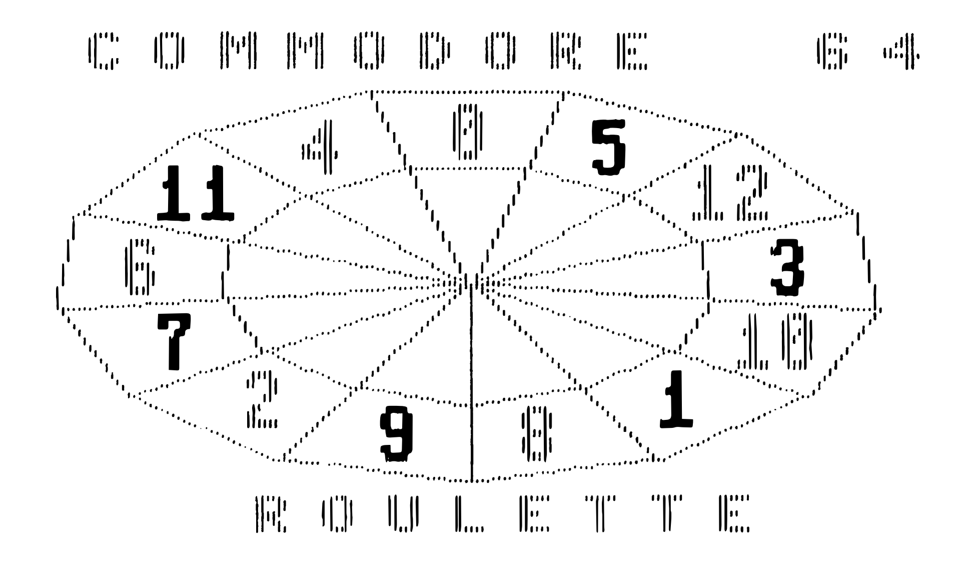


Bild 2.2 : Bildschirmcopy von KESSEL

Der Kessel ist somit in dreizehn Sektoren unterteilt. Die Sektoren wollen wir mit dem LINE-Befehl erstellen, die Zahlen sollen mit dem CHAR-Befehl eingetragen werden und die obere und untere Zeile mit dem TEXT-Befehl. Die rollende Kugel wird durch ein Sprite dargestellt.

Zunächst das Listing:

```
1000 REM ******************************
1010 REM *    VORSPANN                *
1020 REM ******************************
1030 V=53248
1040 REM :
1050 REM --- KOORDINATEN DER ZAHLEN ---
1060 REM :
1070 DIMX(16),Y(16),A(16),F(16)
1080 DATA 71,32,48,1      : REM 0
1090 DATA 109,133,49,3    : REM 1
1100 DATA 34,133,50,2     : REM 2
1110 DATA 129,83,51,3     : REM 3
```

```
1120 DATA 44,38,52,2      : REM 4
1130 DATA 97,38,53,3      : REM 5
1140 DATA 12,83,54,2      : REM 6
1150 DATA 18,111,55,3     : REM 7
1160 DATA 84,146,56,2     : REM 8
1170 DATA 58,146,57,3     : REM 9
1180 DATA 123,111,49,2    : REM 10 (1)
1190 DATA 131,111,48,2    : REM 10 (2)
1200 DATA 18,55,49,3      : REM 11 (1)
1210 DATA 26,55,49,3      : REM 11 (2)
1220 DATA 115,55,49,2     : REM 12 (1)
1230 DATA 123,55,50,2     : REM 12 (2)
1240 REM :
1250 REM ----- KOORDINATEN EINLESEN ---
1260 REM :
1270 FORI=1TO16
1280 READX(I)
1290 READY(I)
1300 READA(I)
1310 READF(I)
1320 NEXT
1330 REM :
1340 REM ----- KOORDINATEN KUGEL -------
1350 REM :
1360 DIMX0(13),Y0(13)
1370 DATA 163,82            : REM 0
1380 DATA 215,88            : REM 5
1390 DATA 267,105           : REM 12
1400 DATA 279,133           : REM 3
1410 DATA 267,161           : REM 10
1420 DATA 239,183           : REM 1
1430 DATA 189,196           : REM 8
1440 DATA 137,196           : REM 9
1450 DATA 89,183            : REM 2
1460 DATA 57,161            : REM 7
1470 DATA 45,133            : REM 6
1480 DATA 65,105            : REM 11
1490 DATA 109,88            : REM 4
1500 REM :
1510 REM ----- KUGEKKOOR. EINLESEN ----
1520 REM :
1530 FORI=1TO13
1540 READ X0(I)
1550 READ Y0(I)
1560 NEXT
1570 REM :
1580 REM ----- KUGEL DEFINIEREN --------
1590 REM :
1600 DESIGN 0,32*64+49152
1610 @.......BBBBBBB..........
1620 @.....BBBBBBBBBBB........
```

```
1630 @....BBBBBBBBBBBBB.......
1640 @...BBBBBBBBBBBBBBB......
1650 @..BBBBBBBBBBBBBBBBB.....
1660 @.BBBBBBBBBBBBBBBBBBB....
1670 @.BBBBBBBBBBBBBBBBBBB....
1680 @BBBBBBBBBBBBBBBBBBBBB...
1690 @BBBBBBBBBBBBBBBBBBBBB...
1700 @BBBBBBBBBBBBBBBBBBBBB...
1710 @BBBBBBBBBBBBBBBBBBBBB...
1720 @BBBBBBBBBBBBBBBBBBBBB...
1730 @BBBBBBBBBBBBBBBBBBBBB...
1740 @BBBBBBBBBBBBBBBBBBBBB...
1750 @.BBBBBBBBBBBBBBBBBBB....
1760 @.BBBBBBBBBBBBBBBBBBB....
1770 @..BBBBBBBBBBBBBBBBB.....
1780 @...BBBBBBBBBBBBBBB......
1790 @....BBBBBBBBBBBBB.......
1800 @.....BBBBBBBBBBB........
1810 @.......BBBBBBB..........
1820 REM ******************************
1830 REM *     HAUPTPROGRAMM            *
1840 REM ******************************
1850 HIRES7,7: MULTI 6,0,2
1860 POKE53280,7
1870 POKE53281,7
1880 X=1
1890 REM :
1900 REM --- TEXTE OBEN UND UNTEN ----
1910 REM :
1920 TEXT 5,5,"COMMODORE  64",1,2,12
1930 TEXT 35,180,"ROULETTE",1,2,12
1940 REM :
1950 REM --- RAHMEN ZEICHNEN ----------
1960 REM :
1970 FORY=0TO2*π-.4STEP2*π/13
1980 LINE 75,100,75+75*SIN(Y),100+75*COS(Y),1
1990 Z=Y+2*π/13
2000 LINE 75+75*SIN(Y),100+75*COS(Y),75+75*SIN(Z),100
                                             +75*COS(Z),1
2010 Y1=75+3/5*(75*SIN(Y))
2020 Y2=100+3/5*(75*COS(Y))
2030 Z1=75+3/5*(75*SIN(Z))
2040 Z2=100+3/5*(75*COS(Z))
2050 LINE Y1,Y2,Z1,Z2,1
2060 NEXT
2070 REM :
2080 REM --- ZAHLEN ZEICHNEN ----------
2090 REM :
2100 FORX=1TO16
2110 CHAR X(X),Y(X),A(X),F(X),3
2120 NEXT
2130 PAUSE1
```

```
2140 MOB SET 0,32,1,0,0
2150 REM :
2160 REM --- KUGEL DEFINIEREN UND -----
2170 REM --- VON MITTE NACH ZERO  ----
2180 REM --- WANDERN LASSEN       -----
2190 REM :
2200 MMOB 0,171,150,163,82,0,255
2210 PAUSE2
2220 XY=1
2230 REM :
2240 REM --- ZUFALLSZAHL FUER ENDE ---
2250 REM --- BESTIMMEN UND KUGEL   ---
2260 REM --- ROLLEN                ---
2270 REM :
2280 ZU=INT(12*RND(6))
2290 FORI=1TO100+ZU
2300 X0=X0(XY)
2310 Y0=Y0(XY)
2320 XY=XY+1
2330 IFXY>13THENXY=1
2340 RLOCMOB 0,X0,Y0,0,I
2350 NEXT
2360 PAUSE10
2370 POKEV+21,0
2380 END
```

Im Vorspann werden zunächst die Daten für die auszugebenden Ziffern am Spielfeld definiert. Da der CHAR-Befehl verwendet werden soll, müssen die X- und Y-Koordinaten, der ASCII-Wert des Zeichens und die Farbe angegeben werden. Bei den Farben soll die Zahl natürlich entsprechend rot oder schwarz eingefärbt sein und die Null wollen wir blau einfärben. Obwohl nur dreizehn Zahlen ausgegeben werden müssen, brauchen wir insgesamt sechzehn Werte, da drei Zahlen je zwei Ziffern besitzen. In den Zeilen 1270 bis 1320 werden die Daten in die entsprechenden Felder eingelesen.

Als nächstes werden die Positionsdaten für die dreizehn Felder bestimmt, die nachher die Bewegung der Kugel simulieren sollen. Der Einfachheit halber springt die Kugel auf einem geraden Weg in das neue Feld. Optisch fällt dies fast gar nicht auf, da ein Kreis mit dreizehn 'Ecken' beschrieben wird. Die Daten für die Kugel sind in der Reihenfolge am Spielfeld angegeben: im Uhrzeigersinn; so, wie später auch die Kugel rollen soll. Auch diese Daten werden in einer FOR...NEXT-Schleife eingelesen.

Dann wird das Sprite für die Kugel definiert, wobei hier zu beachten ist, daß im hochauflösenden Grafikmodus der Wert 49152 ($C000) bei den Blockadressen zu addieren ist.

Im Hauptprogramm wird zunächst die Grafik eingeschaltet und - für die Ausgabe nach Abbruch des Programms - die entsprechenden Werte für Rahmen und Hintergrund gesetzt. Die Zeilen 1920 und 1930 geben die beiden Textzeilen am oberen und unteren Bildschirmrand aus. Die beiden Zahlen vor dem Text geben die X- und Y-Position an, die drei Zahlen dahinter Zeichentyp, Größe und Abstand der Buchstaben. Die letzten Parameter wurden bewußt gleich gewählt. In den Zeilen 1970 bis 2060 wird der Rahmen gezeichnet, dabei wird eine Schleife von 0 bis 2*PI durchlaufen. Dies beschreibt einen Vollkreis. Durch die Schrittweite wird der Vollkreis in dreizehn Teile geteilt. Für jedes Feld wird der entsprechende Radius gezeichnet und anschließend werden zwei Kreisabschnitte, die den inneren und äußeren Kreis des Spielfeldes darstellen sollen, dargestellt. Die Berechnung mag am Anfang etwas kompliziert aussehen, ist es aber nicht, wenn man sich verdeutlicht, daß die beiden Winkelfunktionen Sinus und Cosinus bei entsprechender Parameterisierung auch einen Kreis beschreiben. Etwas übersichtlicher als die ersten beiden LINE-Befehle, aber auch speicherplatzaufwendiger ist der letzte LINE-Befehl, dessen Vorlauf in Zeile 2010 beginnt.

Die bisher gezeichnete Figur kann auch mit dem ARC-Befehl dargestellt werden, wodurch die Berechnungen mit Sin und COS entfallen. Die zeitliche Reihenfolge ist allerdings dann auch geändert. Die Sequenz könnte dann wie folgt aussehen:

```
1950 REM --- RAHMEN ZEICHNEN ----------
1960 REM :
1970 FORY=0TO360STEP360/13
1980 ANGL75,100,Y,75,75,1
1990 NEXT
2000 ARC75,100,0,360,360/13,75,75,1
2010 ARC75,100,0,360,360/13,45,45,1
```

In der Schleife von Zeile 2100 bis Zeile 2120 werden die weiter oben eingelesenen Daten mit dem CHAR-Befehl auf dem Bildschirm ausgegeben.

Mit dem Befehl in Zeile 2140 werden zunächst die grundlegenden Daten für das Sprite der Kugel definiert (vergl. auch Kapitel 3). Mit dem MMOB-Befehl in Zeile 2200 wird

die Kugel mit langsamster Geschwindigkeit vom Mittelpunkt zu der Zero im Spielfeld bewegt. Dies soll den bei Spielbeginn üblichen Vorgang nachbilden. Dann wird in Zeile 2280 eine Zufallszahl zwischen 0 und 12 definiert. Dies soll weiter unten eine der dreizehn Zahlen zufällig auswählen. Damit die Kugel nicht zu kurz rollt, wird in Zeile 2290 der Endwert der Schleife auf 100 plus Zufallszahl festgelegt. In der Variablen XY wird festgehalten, welcher Wert aus den Feldern XO() und YO() zur Berechnung der neuen Position der Kugel herangezogen werden soll. Diese Variable wird innerhalb der Schleife jeweils um eins erhöht und bei Erreichen der vierzehn auf eins zurückgesetzt. Mit dem RLOCMOB-Befehl in Zeile 2340 wird die Kugel nun bewegt, wobei als Geschwindigkeit noch der Schleifenparameter I herangezogen wird, um das Langsamerwerden der Kugel zu simulieren.

2.3 Balkendiagramme

In diesem Kapitel wollen wir kurz auf ein Programm eingehen, das wir schon in Band 3 vorgestellt haben: Balkendiagramme im Mehrfarbenmodus, mit einem dreigeteilten Balken. Dies ist eine übliche Methode, um Prozentveränderungen oder überhaupt zahlenmäßige Veränderungen grafisch darzustellen.

In Anhang 2 finden Sie das Programmlisting, wie wir es in Band 3 besprochen haben.

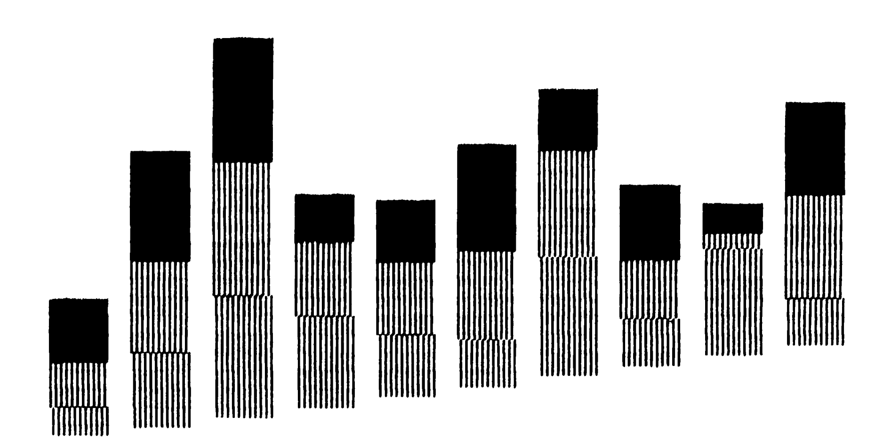


Bild 2.3 : Bildschirmcopy eines Beispiels von BALKEN5

Daß gegenüber dem im Anhang 2 vorgestellten Programm nicht viel geändert werden muß, sehen Sie an folgendem Listing:

```
1000 REM ************************************************
1010 REM *                B A L K E N   5                *
1020 REM ************************************************
1090 COLOUR 11,11
2000 REM ************************************************
2010 REM *     DATEN ERFASSEN UND VORBEREITEN           *
2020 REM ************************************************
2030 INPUT"ANZAHL DER BALKEN";AB
2040 DIMWB(AB,3)
2050 DIMHO(AB,3)
2060 REM----- WERTE DER BALKEN ERFASSEN -----------------
2070 FORI=1TOAB
2080 FORJ=1TO3
2090 PRINTI;"TER BALKEN ";J;"-TER EINTRAG";
2100 INPUTWB(I,J)
2110 WB(I,0)=WB(I,0)+WB(I,J)
2120 NEXT
2130 NEXT
2140 REM----- MAXIMUM BESTIMMEN ------------------------
2150 FORI=1TOAB
2160 IFWB(I,0)>MATHENMA=WB(I,0)
2170 NEXT
2180 REM------ GRAFISCHE AUSGABEDATEN ERFASSEN -----------
2190 INPUT"ABSTAND                    ";A
2200 INPUT"BREITE                     ";BR
2210 INPUT"GRUNDZEILE                 ";GZ
2220 INPUT"MAXIMALE HOEHE (ZEILE)     ";MH
2230 INPUT"ERHOEHUNG                  ";EH
2240 INPUT"BEGINN IN SPALTE           ";BS
2250 REM----- BALKENHOEHEN NORMIEREN --------------------
2260 FORI=1TOAB
2270 HO(I,0)=INT(WB(I,0)*(GZ-MH-EH*(AB-1))/MA+.5)
2280 NEXT
2290 REM----- GROESSE DER TEILBALKEN BESTIMMEN ----------
2300 FORI=1TOAB
2310 FORJ=1TO3
2320 HO(I,J)=HO(I,0)*WB(I,J)/WB(I,0)
2330 NEXT
2340 NEXT
3000 REM ************************************************
3010 REM *     AUSGABE DER BALKEN                       *
3020 REM ************************************************
3030 HIRES 0,0 : MULTI 7,1,8
3040 UX=BS-A
3050 FORI=1TOAB
3060 OX=UX+A
3070 UX=OX+BR
```

```
3080 OY=GZ-HO(I,1)-EH*(I-1)
3090 UY=GZ-EH*(I-1)
3100 BLOCK OX,OY,UX,UY,1
3110 OY=GZ-HO(I,1)-HO(I,2)-EH*(I-1)
3120 UY=GZ-HO(I,1)-EH*(I-1)
3130 BLOCK OX,OY,UX,UY,2
3140 OY=GZ-HO(I,1)-HO(I,2)-HO(I,3)-EH*(I-1)
3150 UY=GZ-HO(I,1)-HO(I,2)-EH*(I-1)
3160 BLOCK OX,OY,UX,UY,3
3170 NEXT
3180 GETA$:IFA$=""THEN3180
3200 END
```

Die Zeilen 1030 bis 1070 entfallen, da die in Band 3 vorgestellten Maschinenprogramme nicht geladen werden müssen (wir gehen davon aus, daß Simon's Basic bereits geladen ist). Die Zeilen 1080 und 1090 werden zu dem COLOUR-Befehl in Zeile 1090 zusammengefaßt, die Zeile 1100 entfällt, da die Adresse des Video-Controllers nicht benötigt wird.

Die Zeilen 2000 bis 2340 sind vollkommen identisch, da sich an der Erfassung und Vorbereitung der Daten nichts ändert.

Die nächste Änderung geschieht in Zeile 3030 wo der GREIN-Befehl durch die Befehle HIRES und MULTI ersetzt werden muß. Die einzigen weiteren Änderungen befinden sich in Zeile 3100, 3130 und 3160 wo jeweils der BLOCK-Befehl der geänderten Syntax angepaßt werden muß, und in Zeile 3190, da die Grafik bei Simon's Basic nicht abgeschaltet werden braucht.

Für diejenigen, die nicht im Besitz von Band 3 sind, kurz noch etwas zur Erfassung der Daten und deren Aufbereitung. Zunächst wird in Zeile 2030 die Anzahl der Balken erfaßt und dementsprechend ein Feld für die Werte der Balken und deren spätere Höhe (in Zeilen) definiert. Dann werden die Werte der einzelnen Balken eingelesen und im Feld 0 der Matrix WB(,) die Summe der drei Teilbalken festgehalten.

In der Schleife von Zeile 2150 bis 2170 wird der größte Wert (Summe) aller Balken ermittelt. Die eingelesenen Daten ab Zeile 2190 dienen der grafischen Aufbereitung und sprechen für sich selbst. Wichtig ist noch die Schleife von Zeile 2260 bis 2280, in der die Höhe der Balken normiert wird. Dies geschieht, indem die vorher eingelesene Grundzeile und die maximale Höhe sowie die mit der Anzahl der Balken multiplizierte Erhöhung voneinander subtrahiert werden. Die Erhöhung gibt an, um wieviel Zeilen jeder weitere Balken höher anfangen soll (mit seiner untersten Zeile) als der vorhergehende.

Aufgrund der Normierung wird in den beiden geschachtelten Schleifen ab Zeile 2300 noch die Höhe in Zeilen für jeden einzelnen Teilbalken berechnet.

In den Zeilen ab 3000 werden dann jeweils für die einzelnen Teilbalken die obere linke und untere rechte Ecke den Variablen OX, UX, OY und UY errechnet und die Blocks ausgegeben.

2.4 Liniendiagramme

Neben den Balkendiagrammen bilden Liniendiagramme eine weitverbreitete Anwendungsform zur grafischen Darstellung unübersichtlicher Zahlenkolonnen. Das beschriebene Programm ist z.B. zur Umsatzdarstellung gut zu verwenden. Die drei möglichen Farben wurden so aufgeteilt, daß eine Farbe für die Rasterung herangezogen wird und die beiden anderen jeweils für eine Umsatzgruppe. Wie in den anderen Fällen auch, ist natürlich die farbige Darstellung mittels der Hardcopyausgaben nicht so gut möglich.

Besonderen Wert wurde darauf gelegt, daß die Daten sowohl vertikal als auch horizontal beliebig gestaucht oder gestreckt werden können. Dies wird auch aus den folgenden Abbildungen deutlich:

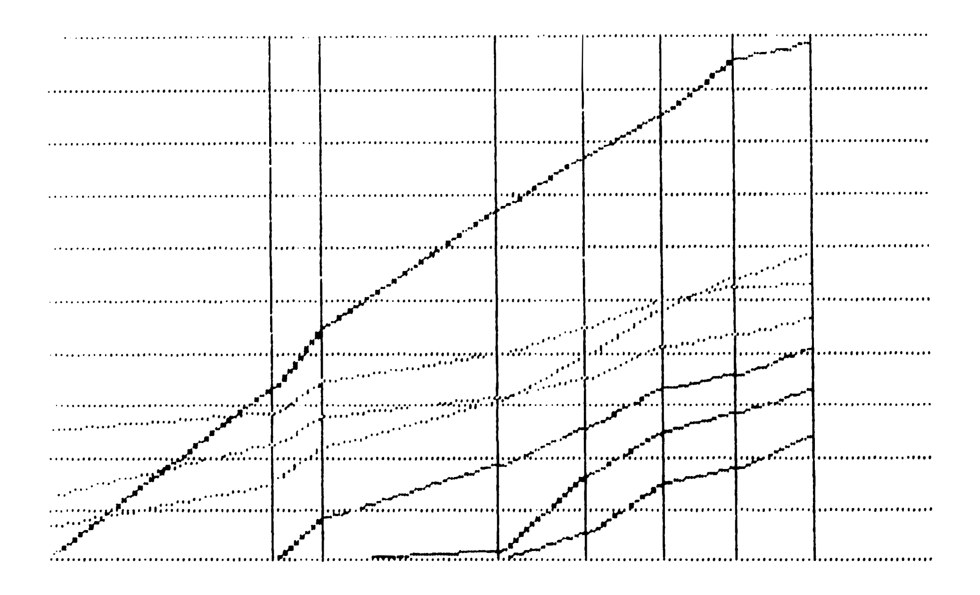


Bild 2.4-1 : Liniendiagramm mit Divisor 1 und Zeitfaktor 1

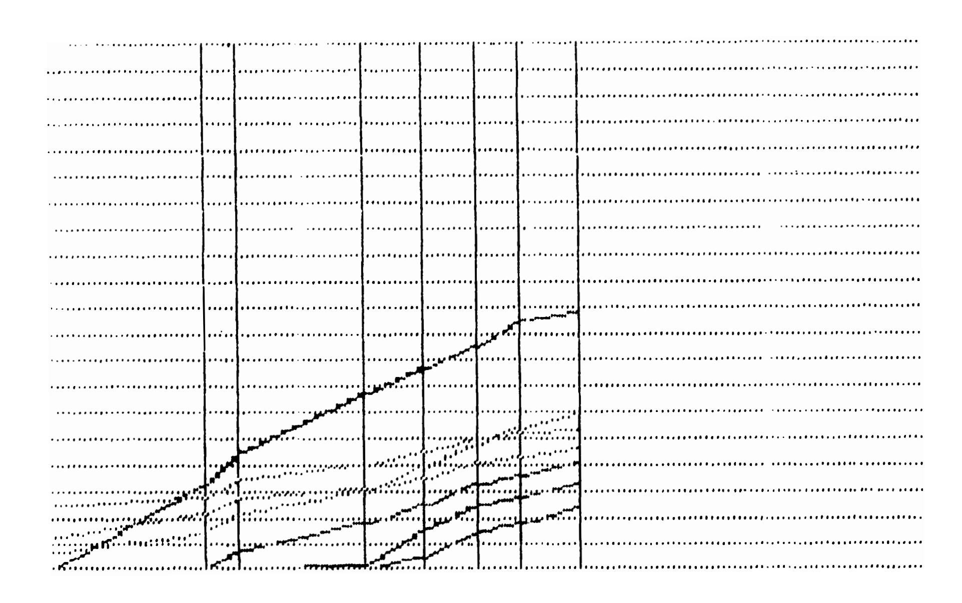


Bild 2.4-2 : Liniendiagramm mit Divisor 2,0 u. Zeitfaktor 0,7

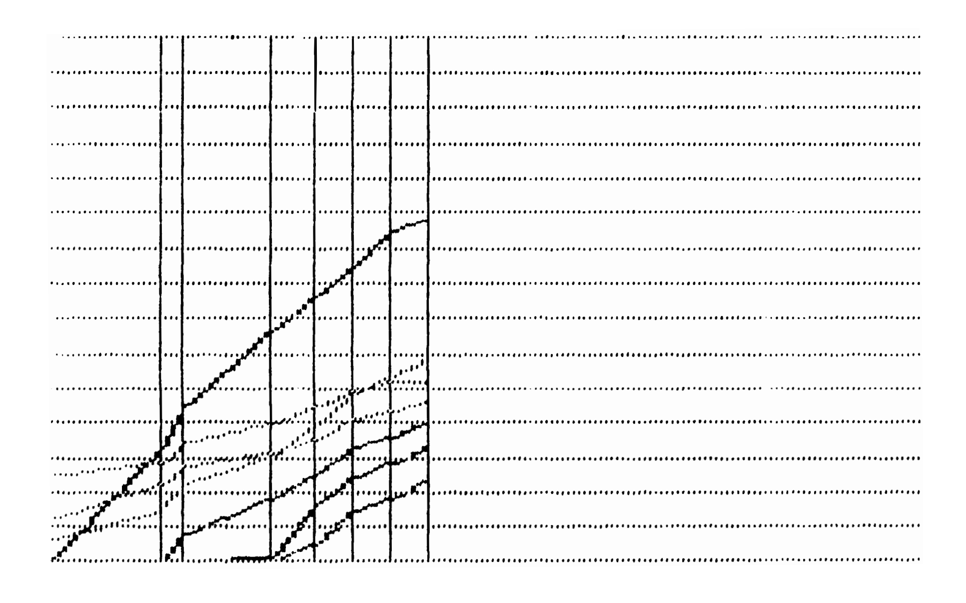


Bild 2.4-3 : Liniendiagramm mit Divisor 1,5 u. Zeitfaktor 0,5

Zunächst jedoch wieder das Programmlisting:

```
1000 REM ******************************
1010 REM *      VORSPANN              *
1020 REM ******************************
1030 REM:
1040 REM --- PARAMETER ---------------
1050 REM:
1060 N=8
1070 DI=1.5
1080 ZF=0.5
1090 REM:
1100 REM --- DATEN -------------------
1110 REM:
1120 DIMZE(N),B(N,2),C(N,4)
1130 DATA0,45,10,36,17,16,14,16
1140 DATA51,57,70,80,90,100,105,106
1150 DATA26,45,56,63,71,83,87,93
1160 DATA3,67,90,134,154,170,191,197
1170 DATA0,1,17,37,51,67,72,82
1180 DATA0,0,0,4,33,50,57,66
1190 DATA0,0,0,0,11,30,36,48
1200 REM:
1210 REM --- DATEN EINLESEN ----------
1220 REM:
1230 FOR I=1 TO N
1240 READZE(I)
1250 ZE(I)=ZE(I)*ZF
1260 NEXT
1270 FOR J=1 TO 2
1280 FOR I=1 TO N
1290 READB(I,J)
1300 NEXT
1310 NEXT
1320 FOR J=1 TO 4
1330 FOR I=1 TO N
1340 READC(I,J)
1350 NEXT
1360 NEXT
1370 REM ******************************
1380 REM *      HAUPTPROGRAMM         *
1390 REM ******************************
1400 HIRES 1,0 : MULTI 6,7,8
1410 REM:
1420 REM --- DATEN NORMIEREN ---------
1430 REM:
1440 FORI=1TON
1450 FORJ=1TO2
1460 B(I,J)=INT(B(I,J)/DI)
1470 NEXT
```

```
1480 FORJ=1TO4
1490 C(I,J)=INT(C(I,J)/DI)
1500 NEXT
1510 NEXT
1520 REM:
1530 REM --- AUSGABESCHLEIFE ----------
1540 REM:
1550 FORI=1TON-1
1560 AX=AX+ZE(I)*.9
1570 EX=EX+ZE(I+1)*.9
1580 REM:
1590 REM --- DATEN AUSGEBEN -----------
1600 REM:
1610 FORJ=1TO2
1620 LINE AX,200-B(I,J),EX,200-B(I+1,J),1
1630 NEXT
1640 FORJ=1TO4
1650 LINE AX,200-C(I,J),EX,200-C(I+1,J),3
1660 NEXT
1670 REM:
1680 REM --- DURCHSCHNITTSWERTE BEST. -
1690 REM:
1700 FORJ=1TO2
1710 DU=DU+B(I,J)
1720 D1=D1+B(I+1,J)
1730 NEXT
1740 FORJ=1TO4
1750 DU=DU+C(I,J)
1760 D1=D1+C(I+1,J)
1770 NEXT
1780 DU=DU/6
1790 D1=D1/6
1800 REM:
1810 REM --- DURCHSCHNITTE UND SENK-  -
1820 REM --- RECHTE SKALIERUNG        -
1830 REM:
1840 LINE AX,200-DU,EX,200-D1,2
1850 LINE EX,0,EX,200,2
1860 NEXT
1870 REM:
1880 REM --- WAAGRECHT SKALIERUNG -----
1890 REM:
1900 FORI=200TO0STEP-20/DI
1910 LINE X,I,160,I,2
1920 NEXT
1930 GETA$:IFA$=""THEN1930
1940 END
```

Neben dem Faktor für die vertikale Streckung (Divisor/DI) und dem Faktor für die horizontale Streckung (Zeitfaktor/ZF) ist auch die Anzahl der zu behandelnden Elemente je Linie variabel (N). Diese Daten werden in den Zeilen 1060 bis 1080 besetzt. Sollten Sie z.B. später einmal mehr Daten haben, und eine horizontale Stauchung notwendig sein, damit alle Daten auf dem Bildschirm Platz finden können, so brauchen Sie nur in Zeile 1080 den Wert hinter ZF zu ändern. Ebenso ist nur in Zeile 1070 der Faktor hinter DI zu ändern, wenn Ihre Umsätze derart in die Höhe schießen, daß Sie den oberen Bildschirmrand erreichen.

Wie Sie an den Abbildungen sehen, sind für die Stichtage, an denen die Werte festgestellt wurden, jeweils senkrechte Linien gezogen. Um dies zu ermöglichen, muß natürlich noch der Abstand der Linien angegeben werden. Die Daten dazu befinden sich in Zeile 1130, wobei die Daten Ihren Wünschen entsprechend Tage, Wochen oder Monate angeben können. Vorher wurden jedoch noch die benötigten Felder dimensioniert: ZE() für die Zeitabstände, B() für die erste Produktgruppe mit zwei Elementen und C() für die zweite Produktgruppe mit vier Elementen.

Ab Zeile 1140 befinden sich die Daten für die Produkte. Dabei ist für jedes Produkt eine Zeile vorgesehen, so daß Sie die Erweiterung Ihrer Daten jeweils durch Anhängen des neuesten Ergebnisses an die Zeile erreichen können.

Ab Zeile 1230 werden die Daten eingelesen, hier zunächst die Zeitabstände die noch mit dem Zeitfaktor multipliziert werden, d.h., daß Sie in den DATA-Zeilen die originären Daten eingeben können, wie Sie bei Ihnen anfallen. Durch die Vorgehensweise beim Eintragen der Daten (je Produkt eine Zeile) müssen natürlich die Schleifen ab Zeile 1270 etwas anders gestaltet werden als normal üblich. Die äußere Schleife läuft jeweils für die Produktgruppe und die innere für die eingegebenen Elemente. Ob Sie Ihre Daten in Stück oder TDM angeben, bleibt auch Ihnen überlassen.

Nachdem in Zeile 1400 die Grafik eingeschaltet wurde, wird in den Schleifen ab Zeile 1440 die Normierung der Daten durchgeführt. Die Ausgabeschleife für die Linien beginnt in Zeile 1550, wobei zunächst die X-Koordinaten für die Linien festgelegt werden. Die Ausgabe geht so vor, daß zunächst für einen Zeitbereich alle Linien ausgegeben werden, deshalb auch zunächst die Berechnung der X-Werte. Ab Zeile 1610 befinden sich die Ausgabeschleifen für die beiden Produktgruppen.

Anschließend wird ab Zeile 1700 noch der Durchschnittswert

gebildet, der natürlich für beide Endpunkte der Linien berechnet werden muß. Achten Sie darauf, daß in den Zeilen 1780 und 1790 jeweils die '6' durch einen entsprechenden Wert ersetzt werden muß, wenn Sie die Darstellung für mehr oder weniger Produkte verwenden.

In Zeile 1840 wird die Linie für den Durchschnitt angegeben, auch in der Farbe '2' für die Skalierung. In Zeile 1850 wird noch die senkrechte Linie gezogen, was man an den Y-Koordinaten (0,200) leicht sieht. Nach Abschluß der Datenausgabe und der zeitlichen Skalierung kann die waagerechte Skalierung durchgeführt werden (ab Zeile 1900). In Zeile 1900 kann die Zahl '20' noch durch entsprechende Werte ersetzt werden. Sie bedeutet, daß je Bildschirmzeile 20 Einheiten der originär eingegebenen Daten aufgetragen werden. Da die Daten mit DI normiert wurden, muß natürlich auch die Schrittweite innerhalb der Schleife mit DI normiert werden.

2.5 Zeichnen mit Joystick und Paddles

Die Befehle JOY und POT(0) sowie POT(1) ermöglichen eine komfortable Joystick- bzw. Paddle-Abfrage. Da jeweils zwei Paddles an einem Stecker sitzen, gibt es folglich für jedes Paddle einen eigenen Befehl. Die Abfrage von Joystick und Paddles ist nur an Control-Port 2 möglich. Gehen wir zunächst kurz auf den Joystick ein. Der am Joystick anliegende Wert, kann sowohl einer Variablen zugewiesen werden, wie unser erstes kurzes Beispiel zeigt:

```
100 P=JOY
110 PRINTP
120 GOTO100
```

als auch bei der Eingabe direkt verwertet werden, wie folgende Zeilen zeigen:

```
200 PRINTJOY
210 GOTO200
```

Sie sollten diese kurzen Testprogramme ruhig einmal laufen lassen, und dabei den Joystick bewegen, um sich die Arbeit des Rechners dabei besser klarzumachen.

Ähnliches gilt auch für die Paddles. Zunächst die Zuweisung für die Variablen:

```
100 P0=POT(0)
110 P1=POT(1)
```

```
120 PRINTP1,P2
130 GOTO100
```

und nun die direkte Verarbeitung:

```
200 PRINTPOT(0),POT(1)
210 GOTO200
```

2.5.1 Hochauflösende Grafik mit dem Joystick

Zunächst wollen wir ein Programm vorstellen, bei dem Sie in normaler hochauflösender Grafik mit dem Joystick zeichnen können.

```
100 HIRES6,7
110 X=160
120 Y=100
130 P=JOY
140 IF P=1 OR P=129 THEN Y=Y-1
150 IF P=2 OR P=130 THEN Y=Y-1 : X=X+1
160 IF P=3 OR P=131 THEN X=X+1
170 IF P=4 OR P=132 THEN Y=Y+1 : X=X+1
180 IF P=5 OR P=133 THEN Y=Y+1
190 IF P=6 OR P=134 THEN Y=Y+1 : X=X-1
200 IF P=7 OR P=135 THEN X=X-1
210 IF P=8 OR P=136 THEN Y=Y-1 : X=X-1
240 IF P>127 THEN ZF=1 : ELSE: ZF=0
245 GETA$
246 IFA$="P"THEN PAINT X,Y,1 : A$=""
250 PLOT X,Y,ZF
260 GOTO130
```

Gerade Linien (senkrecht, waagerecht und diagonal) sind dabei kein Problem. Schwierigkeiten gibt es allerdings, wenn man schöne Kurven zeichnen will. So ist auch die nachfolgende Abbildung kein großes Meisterwerk. Für Kurven müssen Sie immer zwischen zwei Joystickrichtungen hin und her pendeln.

In dem Programm wollen wir den 'Feuerknopf' dazu benutzen, entweder an der aktuellen Bildschirmposition einen Punkt zu setzen (Feuerknopf gedrückt) oder zu löschen (Feuerknopf nicht gedrückt).

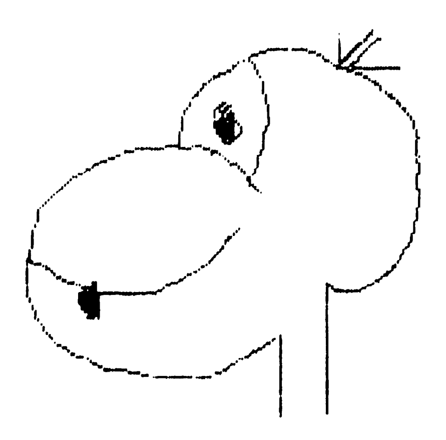


Bild 2.5-1 : Figur mit Joystick gezeichnet

Zunächst wird die hochauflösende Grafik eingeschaltet und die Zeichenposition auf die Mitte des Bildschirms festgelegt (Zeile 110 und 120). Um nicht jeweils die JOY-Abfrage durchführen zu müssen, wird der aktuelle Wert zunächst an die Variable P übergeben. Damit ist auch sichergestellt, daß nicht während des Ablaufes eine andere Joystickposition überprüft wird.

Zunächst wird in den Zeilen 140 bis 210 die Richtung des Joystickes festgestellt, und die X- und Y-Koordinaten werden entsprechend verändert. Ist der Feuerknopf gedrückt (P gößer als 127), so wird die Hilfsvariable ZF auf 1 gesetzt, andernfalls auf 0.

In Zeile 245 wird noch ein Zeichen von der Tastatur eingelesen, um auch das Ausfüllen einer umrandeten Fläche zu gestatten. In Zeile 246 wird abgeprüft, ob der Tastendruck ein P ist, wodurch der PAINT-Befehl aufgerufen wird. Anschließend wird die Hilfsvariable A$ wieder zurückgesetzt. Achtung: Es ist keine Warteschleife programmiert, so daß Sie zu jedem beliebigen Zeitpunkt 'P' drücken können.

In Zeile 250 wird der eigentliche Punkt ausgegeben. Durch die Syntax beim PLOT-Befehl wird über die Hilfsvariable ZF bestimmt, ob der Punkt gesetzt oder gelöscht wird. Dann springt das Programm zur erneuten Abfrage des Joystickwertes.

Sind Sie mit dem Zeichnen fertig, so drücken Sie einfach

die STOP-Taste. Danach können Sie mit dem COPY-Befehl die Grafik auf dem Drucker ausgeben.

2.5.2 Mehrfarben-Grafik mit dem Joystick

```
100 HIRES1,11
105 MULTI 12,8,6
106 EF=1
110 X=80
120 Y=100
130 P=JOY
140 IF P=1 OR P=129 THEN Y=Y-1
150 IF P=2 OR P=130 THEN Y=Y-1 : X=X+1
160 IF P=3 OR P=131 THEN X=X+1
170 IF P=4 OR P=132 THEN Y=Y+1 : X=X+1
180 IF P=5 OR P=133 THEN Y=Y+1
190 IF P=6 OR P=134 THEN Y=Y+1 : X=X-1
200 IF P=7 OR P=135 THEN X=X-1
210 IF P=8 OR P=136 THEN Y=Y-1 : X=X-1
220 GETA$
230 IFA$="1" THEN EF=1
232 IFA$="2" THEN EF=2
233 IFA$="3" THEN EF=3
234 IFA$="4" THEN EF=4
235 PRINTA$,P,ZF,EF
240 IF P>127 THEN ZF=EF : ELSE: ZF=0
250 PLOT X,Y,ZF
260 GOTO130
```

Das Programm aus dem vorigen Kapitel brauchen wir für den Mehrfarbenmodus nur leicht zu ändern. Zunächst muß nach dem Einschalten der Grafik mit dem MULTI-Befehl festgelegt werden, welche Farben die Zeichnung haben soll. Per Programm wird die erste Zeichenfarbe (Variable EF) auch mit der ersten Farbe im MULTI-Befehl gesetzt. Zu beachten ist, daß in Zeile 110 der X-Wert für die Bildschirmmitte auf 80 gesetzt werden muß, da die Auflösung nur noch 160 Zeichenpunkte vertikal beträgt.

Die Zeilen 120 bis 210 sind identisch dem Programm im vorigen Kapitel. Die Abfrage für den PAINT-Befehl wurde in diesem Programm ausgelassen. Dafür werden die vier Tasten '1', '2', '3' und '4' abgefragt und jeweils die Hilfsvariable für die zu zeichnende Farbe entsprechend umgesetzt (EF). Zur Kontrolle werden die Parameter in Zeile 235 noch am Bildschirm ausgegeben. Solange die Grafik eingeschaltet ist, werden diese natürlich nicht angezeigt. Brechen Sie das Programm mit der STOP-Taste ab, so sehen Sie die letz-

ten Ausgaben noch auf dem Bildschirm.

In Zeile 240 wird wieder die zu zeichnende Farbe festgelegt. Dazu wird in diesem Programm die Hilfsvariable ZF mit dem Wert von EF besetzt. In Zeile 250 erfolgt wieder die Ausgabe und in Zeile 260 wird zur erneuten Joystickabfrage übergegangen.

2.5.3 Mehrfarben-Grafik mit Joystick und PAINT

In diesem kurzen Kapitel wollen wir das Programm aus dem letzten Kapitel dahingehend ändern, daß umrandete Bildschirmgebiete auch mit einer Farbe nach Wahl gefüllt werden können. Bei der Handhabung des Programms können Sie also mit dem Joystick in ein umrandetes Gebiet gehen, anschließend die Farbe wählen und dann 'P' drücken, wonach das angewählte Gebiet mit der gewünschten Farbe ausgefüllt wird.

```
100 HIRES1,11
105 MULTI 12,8,6
106 EF=1
110 X=80
120 Y=100
130 P=JOY
140 IF P=1 OR P=129 THEN Y=Y-1
150 IF P=2 OR P=130 THEN Y=Y-1 : X=X+1
160 IF P=3 OR P=131 THEN X=X+1
170 IF P=4 OR P=132 THEN Y=Y+1 : X=X+1
180 IF P=5 OR P=133 THEN Y=Y+1
190 IF P=6 OR P=134 THEN Y=Y+1 : X=X-1
200 IF P=7 OR P=135 THEN X=X-1
210 IF P=8 OR P=136 THEN Y=Y-1 : X=X-1
220 GETA$
230 IFA$="1" THEN EF=1
232 IFA$="2" THEN EF=2
233 IFA$="3" THEN EF=3
234 IFA$="4" THEN EF=4
235 IFA$="P" THEN PAINT X,Y,EF : EF=4
236 PRINTA$,P,ZF,EF
240 IF P>127 THEN ZF=EF : ELSE: ZF=0
250 PLOT X,Y,ZF
260 GOTO130
```

Dazu ist zunächst (Zeile 235) zu prüfen, ob die Taste 'P' gedrückt wurde. Anschließend ist der PAINT-Befehl mit den entsprechenden Koordinaten und der gewünschten Farbe (EF) durchzuführen. Beachten Sie bitte, daß die Farbe nach jedem PAINT-Befehl automatisch auf '4' umgeschaltet wird.

Der Rest des Programmes ist identisch. Hier noch eine kleine Abbildung einer imaginären Landkarte:

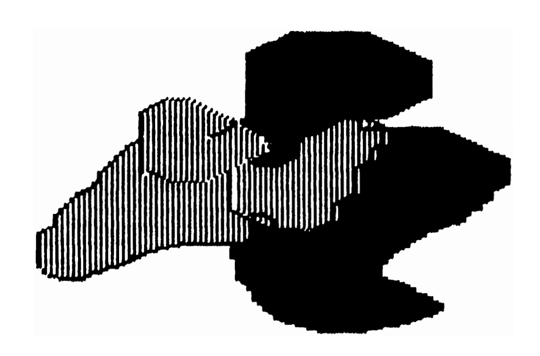


Bild 2.5-2 : 'Landkarte' mit Joystick und PAINT

2.5.4 Joystick, PAINT und 16 Farben

```
100 COLOUR0,0
101 CSET2
105 MULTI 1,2,3
106 EF=1
110 X=80
120 Y=100
130 P=JOY
140 IF P=1 OR P=129 THEN Y=Y-1
150 IF P=2 OR P=130 THEN Y=Y-1 : X=X+1
160 IF P=3 OR P=131 THEN X=X+1
170 IF P=4 OR P=132 THEN Y=Y+1 : X=X+1
180 IF P=5 OR P=133 THEN Y=Y+1
190 IF P=6 OR P=134 THEN Y=Y+1 : X=X-1
200 IF P=7 OR P=135 THEN X=X-1
210 IF P=8 OR P=136 THEN Y=Y-1 : X=X-1
220 GETA$
230 IFA$="1" THEN EF=1
```

```
232 IFA$="2" THEN EF=2
233 IFA$="3" THEN EF=3
234 IFA$="4" THEN EF=4
235 IFA$="P" THEN PAINT X,Y,EF : EF=4
236 IFA$="L" THEN GOSUB 500
240 IF P>127 THEN ZF=EF : ELSE: ZF=0
250 PLOT X,Y,ZF
260 GOTO130
500 GETL$:IFL$=""THEN500
510 L=VAL(L$)
520 IFL<0ORL>4THEN500
530 IF L=0 THEN HI COL : RETURN
540 ON L GOTO 550,560,570,580
550 LOW COL 4,5,6 : RETURN
560 LOW COL 7,8,9 : RETURN
570 LOW COL 10,11,12 : RETURN
580 LOW COL 13,14,15 : RETURN
```

Auch dieses Programm entspricht wieder im großen und ganzen den vorher vorgestellten. Geändert wurde in Zeile 100 und 101 der HIRES-Befehl. Zunächst wurde er durch den COLOUR-Befehl ersetzt, der Rahmen- und Hintergrundfarbe setzt. Anschließend wurde eine andere Grafik mit dem CSET2-Befehl wieder zurück auf den Bildschirm geholt. Als Farben im MULTI-Befehl (der Hintergrund ist schwarz = Farbe 0) wurden einfach die ersten drei Farben herangezogen. Die Zeilen bis 235 sind schon bekannt. In Zeile 236 wird abgefragt, ob an der Tastatur ein 'L' gedrückt wurde; wenn ja, wird in das Unterprogramm ab Zeile 500 verzweigt. Neu sind dann auch nur noch die Zeilen von 500 bis 580. Zunächst wird eine weitere Taste von der Tastatur abgefragt, und in der Variablen L der Zahlwert festgelegt. Zeile 520 beinhaltet eine Plausibilitätsprüfung, ob ein kleinerer Wert als null, oder ein größerer Wert als vier vorliegt. Beachten Sie bitte, daß die Eingabe eines Buchstabens auch auf den Wert null führt. Mit der Wahl der Zahltasten 0 bis 4 können Sie nun weitere Farben auswählen. Geben Sie eine Null ein, so wird auf die ursprünglich im MULTI-Befehl angegebenen Farben zurückgeschaltet. In allen anderen Fällen wird in Zeile 540 zu den weiteren Zeilen verzweigt, die jeweils ein Dreierpaar an Farben angeben. Der Einfachheit halber wurden hier die Farben in ihrer numerischen Reihenfolge herangezogen. Da jede angesprungene Zeile das Unterprogramm beendet und vorher eine Plausibilitätsprüfung durchgeführt wurde, braucht das Unterprogramm selbst nicht mit einem RETURN beendet zu werden.

Wenn Sie nun zeichnen und Ihre Gebiete entsprechend einfärben, werden Sie einige Male feststellen, daß die Gren-

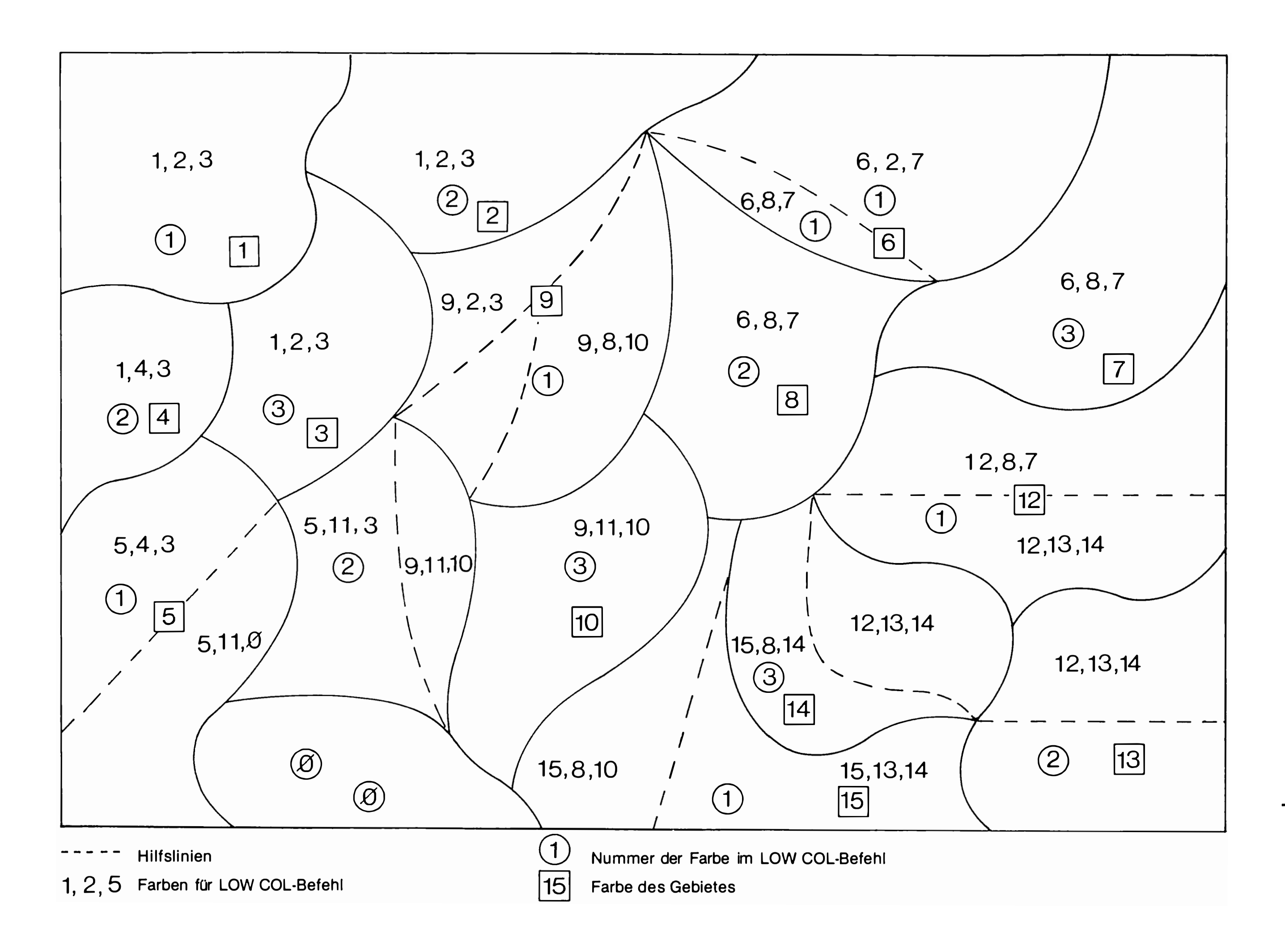
zen treppenförmig eingefärbt werden. Dazu noch ein bißchen über die Speicherung von Farbgrafiken. Wie wir in früheren Bänden schon festgestellt haben, können bis zu drei Farben (und die Hintergrundfarbe) vom Rechner vergeben werden. Dies gilt aber nicht für den gesamten Bildschirm, sondern es kann jedes Zeichen - wie es im normalen Textmodus als Matrix aus 8*8 Punkten erscheint - drei verschiedene Farben bekommen. Bei ungeschickter Wahl der Farben für Ihre Gebiete ergeben sich somit die 'Treppen', dort wo Ihre Linie nicht genau an der Grenze eines Zeichens verläuft.

Dies liegt daran, daß der LOW COL-Befehl immer alle Farben innerhalb dieses Zeichens umbesetzt. Bei geschickter Farbangabe beim LOW COL-Befehl kann man diesen Umstand aber umgehen. Das letzte Programm sollten Sie sich dazu so umschreiben, daß Sie zwar die Farben über den LOW COL-Befehl ändern können, aber bei der Eingabe alle drei Farben selbst über Tastatur bestimmen können. Sinnvoll wäre dabei auch eine zusätzliche Textausgabe am unteren Bildschirmrand, die die gerade aktuellen Farben anzeigt, damit Sie nicht versehentlich falsche Farben wählen. In der nachfolgenden Abbildung haben wir Ihnen die Möglichkeit des Einfärbens mit mehr als drei Farben an einer imaginären Landkarte dargestellt. Die im Kreis dargestellte Zahl gibt jeweils die Farbposition im LOW COL-Befehl an (1, 2 oder 3), die Zahl im Quadrat die Farbnummer für dieses Gebiet, und die Zahlen darüber die jeweils im LOW COL-Befehl einzugebenden Farben.

Die Hilfslinien innerhalb der einzelnen Gebiete müssen so gewählt werden, daß sie jeweils nur zwei weitere Nachbarn haben, wenn Sie die Untergebiete betrachten. Die Farbnummer im LOW COL-Befehl muß innerhalb des gesamten Gebietes natürlich einheitlich sein. Wenn Sie in einem Gebiet die Hintergrundfarbe wählen, so können hier auch drei Nachbarn vorhanden sein.

Da mittlerweile das Vierfarbenproblem mathematisch gelöst wurde, kann man auch sagen, daß man die gesamte 'Landkarte' mit vier Farben einfärben kann, egal wie die Landkarte aussieht und ohne, daß benachbarte Gebiete die gleiche Farbe haben müssen.

Wer will kann sich auch ein Automatikprogramm schreiben, das die Farben in den benachbarten Gebieten jeweils mit PEEK im Farb-RAM (ab $D800) und Video-RAM (ab $C000) ausliest und die Färbung dann automatisch durchführt.



1, 2, 3
1
1
1, 2, 3
2
2
6, 2, 7
1
6
6,8,7
1
6, 8, 7
3
7
1, 4, 3
2
4
1, 2, 3
3
3
9, 2, 3
9
9, 8, 10
1
6, 8, 7
2
8
12, 8, 7
12
1
12, 13, 14
5, 4, 3
1
5
5, 11, Ø
5, 11, 3
2
9, 11, 10
9, 11, 10
3
10
15, 8, 14
3
14
12, 13, 14
12, 13, 14
Ø
Ø
15, 8, 10
1
15, 13, 14
15
2
13
Hilfslinien
1, 2, 5 Farben für LOW COL-Befehl
1 Nummer der Farbe im LOW COL-Befehl
15 Farbe des Gebietes

2.5.5 Hochauflösende Grafik mit Paddles

Auch mit Paddles lassen sich Formen in hochauflösender Grafik darstellen, wenn man je ein Paddle für die X- und Y-Richtung hernimmt. Dies ist zwar etwas mühsamer und erfordert mehr Geschick, aber durchaus realisierbar. Wir haben dieses Beispiel auch gewählt, um Ihnen die Bewegungsmöglichkeiten mit Paddles auf dem Bildschirm darzustellen. Da alle anderen Programmstücke analog den bisher beschriebenen Unterkapiteln von Kapitel 2.5 sind, wollen wir hier nur das Programm für die einfache hochauflösende Grafik darstellen.

```
100 HIRES0,7
102 X=10
104 Y=10
110 P0=POT(0)
120 P1=POT(1)
125 XA=X
130 IFP0>170THEN X=X-1
132 IFP0<85THEN X=X+1
140 IFX>319THENX=X-319
145 IFX<10THENX=10
146 YA=Y
150 IFP1>170THEN Y=Y-1
152 IFP1<85THEN Y=Y+1
160 IFY>199THENY=Y-199
165 IFY<10THENY=10
166 LINE XA-10,YA,XA-5,YA,2
167 LINE XA+10,YA,XA+5,YA,2
168 LINE XA,YA-10,XA,YA-5,2
169 LINE XA,YA+10,XA,YA+5,2
170 LINE X-10,Y,X-5,Y,2
180 LINE X+10,Y,X+5,Y,2
190 LINE X,Y-10,X,Y-5,2
200 LINE X,Y+10,X,Y+5,2
202 GETZ$
203 IFZ$=""THEN205
204 Z=VAL(Z$)
205 PLOT X,Y,Z
210 GOTO110
```

Der Beginn ist auch analog den vorher angesprochenen Programmen, zunächst wird die hochauflösende Grafik eingeschaltet und die Anfangsposition festgelegt. Sie befindet sich bei den Koordinaten (10,10). Dann werden die beiden Paddles abgefragt, und der alte X-Wert der Variablen XA wird zwischengespeichert. Die Zeilen 130 bis 145 beinhal-

ten die Positionierung der X-Koordinaten und die Zeilen 146 bis 165 die entsprechenden Umsetzungen für die Y-Koordinaten, mit Zwischenspeicherung des alten Wertes in YA.

Da wir den aktuellen Punkt durch je zwei senkrechte und waagerechte Linien anzeigen wollen, darf die X-Koordinate nicht kleiner als 10 werden. Die Abfrage der Paddlewerte (die Paddles sind nichts anderes als ein Potentiometer) wird so geschaltet, daß bei den Werten im unteren Drittel der aktuelle Punkt nach rechts und bei dem oberen Drittel der aktuelle Punkt nach links bewegt wird. Im mittleren Drittel erfolgt keine Bewegung. Dadurch ist keine Geschwindigkeitsänderung bei der Bewegung des Punktes gegeben. Dies ist für das Zeichnen von ununterbrochenen Linien unerlässlich. Falls Sie gestrichelte Linien haben wollen, können Sie die Potentiometereinstellung (zwischen 0 und 255) auch anders einteilen und in bestimmten Bereichen den Punkt jeweils um zwei und mehr Bildschirmpunkte wandern lassen.

Bei der einfachen Grafik bedeutet die '2' als Zeichenfarbe, daß der jeweilige Punkt invertiert wird. Dies muß bei der Darstellung des Hilfskreuzes so sein, damit bei der Koordinatenanzeige nicht ein bereits gezeichnetes Bild zerstört wird. Am besten versteht man das Programm durch Ausprobieren.

Die Zeilen ab 202 dienen zum Umsetzen der Zeichenfarbe und zum Setzen des eigentlichen Punktes.

2.6 Neue Zeichen mit DESIGN

Im letzten Kapitel über Grafik-Befehle wollen wir auf die Änderung des Zeichensatzes eingehen. Zuständig sind dafür - ähnlich wie bei den Sprites - die Befehle DESIGN2 und DESIGN3. DESIGN2 ändert dabei Zeichen im Normalmodus und DESIGN3 ist der entsprechende Befehl für den Mehrfarben-Modus.

In den Unterprogrammen wird der DESIGN-Befehl etwas mißbraucht, da als Parameter noch die Stelle am Bildschirm (in Zeilen und Spalten) angegeben wird. Zunächst das Listing:

```
100 HIRES6,7:REM ZEICHENPOSITIONIEREN IN HIRES
110 COLOUR6,15
200 CH=1:Z=5:S=7:GOSUB1000
```

```
210 CH=1:Z=10:S=7:GOSUB1000
220 CH=2:Z=10:S=27:GOSUB1000
300 PAUSE10
310 MULTI7,4,5
320 CH=1:Z=5:S=7:GOSUB2000
330 Z=7:FORS=0TO38STEP2:GOSUB2000:NEXT
340 PAUSE10
500 REM  NEUE ZEICHEN FUER TEXTMODUS
510 CSET0: REM GRAFIK-MODUS AUS
520 MEM :REM ALTEN ZEICHENSATZ NACH $E000
530 Z=0 :REM ZUM ZEICHENDEFINIEREN
540 CH=1:S=0:GOSUB1000 : REM NEUER KLAMMERAFFE
550 CH=2:S=1:GOSUB1000 : REM NEUES A
560 CH=2:S=6:GOSUB2000 : REM NEUES F
570 CH=1:S=7:GOSUB2000 : REM NEUES G
580 POKEGRAPHICS+33,1 : REM HINTERGRUNDF. 0
590 POKEGRAPHICS+34,0 : REM HINTERGRUNDF. 1
600 POKEGRAPHICS+35,2 : REM HINTERGRUNDF. 2
610 POKEGRAPHICS+22,PEEK(GRAPHICS+22)OR16 : REM MCM
620 PRINT"█@ABCDEFG":REM IN CTRL-GELB (NR.7)
630 PRINT
640 PRINT"█@ABCDEFG":REM IN  C=-GELB (NR.15)
650 PRINT"█■@ █@ █F █G"
660 PAUSE10
700 REM GESAMTEN ZEICHENSATZ AUF BILDSCHIRM BRINGEN
710 FCOL0,0,40,25,7:REM FARB-RAM BESCHREIBEN
720 POKEGRAPHICS+22,200 : REM MCM AUS
730 CSET2: REM ZEICHENSATZ (NORMALE ZEICHEN) ZEIGEN
740 PAUSE10
750 MULTI0,1,15: REM JETZT MULTI-ZEICHEN MEHRFARBIG
760 PAUSE10
770 NRM
780 END
1000 CGOTO(1000+100*CH)
1100 DESIGN2,$E000+320*Z+8*S
1110 @...B....
1120 @..BBB...
1130 @.BBBBB..
1140 @BBBBBBB.
1150 @BBBBBBB.
1160 @BBBBBBB.
1170 @BBBBBBB.
1180 @BBBBBBB.
1190 RETURN
1200 DESIGN2,$E000+320*Z+8*S
1210 @BB....BB
1220 @.BB..BB.
1230 @..BBBB..
1240 @..BBBB..
1250 @..BBBB..
1260 @.BB..BB.
```

```
1270 @BB....BB
1280 @BB....BB
1290 RETURN
1300 DESIGN2,$E000+320*Z+8*S
1390 RETURN
2000 CGOTO(2000+100*CH)
2100 DESIGN3,$E000+320*Z+8*S
2110 @BBBC
2120 @BBCC
2130 @BCCC
2140 @CCCC
2150 @DCCC
2160 @DDCC
2170 @DDDC
2180 @DDDD
2190 RETURN
2200 DESIGN3,$E000+320*Z+8*S
2210 @BBBB
2220 @BBBB
2230 @BBBB
2240 @CCCC
2250 @CCCC
2260 @CCCC
2270 @DDDD
2280 @DDDD
2290 RETURN
```

Mit dem Rahmenunterprogramm ab Zeile 1000 können mehrere Zeichen im Normalmodus definiert werden. Entscheidend ist dazu in Zeile 1000 der CGOTO-Befehl, der in Simon's Basic auch Variablen als Sprungziel gestattet. In diesem Falle wird er dazu verwendet, aufgrund der Variablen CH in die entsprechenden Zeilennummern für die Zeichen (1100, 1200, ...) zu verzweigen.

Zunächst wird in Zeile 1100 ein einfarbiges Zeichen neu definiert. Da sich der RAM-Bereich für den Zeichensatz und die hochauflösende Grafik überlagern, gibt der zweite Parameter im DESIGN-Befehl im Textmodus eine Adresse innerhalb des Zeichensatzes an und im Grafikmodus eine Stelle am Bildschirm. Die angegebene Lokalisierung bezieht sich in unseren Unterprogrammen auf den Bildschirm, da wir die Zeichen sofort sichtbar machen wollen. Wenn Sie nur in den Zeichensatz schreiben wollen, so ersetzen Sie den Term für die Zeile durch Null und geben Sie in der Variablen S die Nummer des Zeichens an. Dies könnte dann wie folgt aussehen: DESIGN2,$E000+8*S oder DESIGN2,$E000+8*40, wenn Sie das 40. Zeichen im Zeichensatz ändern wollen.

Die Definition des Zeichens geschieht ähnlich den Sprites und braucht hier nicht näher erläutert zu werden.

Ab Zeile 2000 befinden sich die entsprechenden Unterprogramme für mehrfarbige Zeichen. Hierzu ist noch zu bemerken, daß mehrfarbige und einfarbige Zeichen, die mit dem DESIGN-Befehl erstellt wurden im Textmodus gemischt werden können.

Im Hauptprogramm schalten wir zunächst die hochauflösende Grafik (Normalmodus) ein und setzen die Farben für Hintergrund und Rahmen. Ab Zeile 200 werden die entsprechenden Parameter für die Unterprogramme ab Zeile 1000 gesetzt, um einige Beispiele zu erhalten. Ab Zeile 300 wird ähnliches für mehrfarbige Zeichen durchgeführt.

In Zeile 500 wird zunächst die Grafik ausgeschaltet und der alte Zeichensatz wird in den Grafikspeicher gelegt. Dann werden einige Zeichen im Zeichensatz geändert (Zeile 540 bis 570). Die Zeilen 580 und 600 dienen dem Setzen der Hintergrundfarben 0, 1 und 2, da diese allen Zeichen gemeinsam sind. In Zeile 610 wird noch der Multi-Colour-Modus eingeschaltet, was uns mehrfarbige Textzeichen erlaubt, jedoch gleichzeitig nur noch acht verschiedene Farben zuläßt. Ab Zeile 620 befinden sich noch zwei Beispielzeilen, die aufzeigen sollen, daß der Mehrfarbenmodus nur in Erscheinung tritt, wenn die gewählte Farbe bzw. ihre Nummer größer als 7 ist (zweite Beispielzeile).

Ab Zeile 700 wird gezeigt, wie man auf einfache Weise den gesamten Zeichensatz am Bildschirm anzeigen kann. Durch den MEM-Befehl wurde der Zeichensatz bereits in den Grafikspeicher kopiert. Das Farb-RAM kann bequem mit FCOL beschrieben werden. Wird anschließend der Multi-Colour-Modus ausgeschaltet und die hochauflösende Grafik eingeschaltet, so erscheinen die Zeichen als normale Zeichen, da die im FCOL-Befehl angegebene Farbe kleiner als 8 ist.

Die mehrfarbig definierten Zeichen erscheinen richtig, wenn mit MULTI 0,1,15 einerseits das Farb-RAM mit der Farbe 15 (hier:gelb) beschrieben wird und andererseits der Mehrfarben-Modus eingeschaltet wird.

Am Ende des Programmes wird der veränderte Zeichensatz mit NRM wieder abgeschaltet.

3

Sprite-Befehle am Beispiel RAKETE 5

3. Sprite-Befehle am Beispiel RAKETE

In den Bänden 1 und 3 haben wir bereits die Möglichkeiten der Sprite-Programmierung besprochen, wobei wir in Band 3 dazu vier zusätzliche Basic-Befehle entwickelt haben. Mit diesem Kapitel wollen wir gleich zwei Aufgaben erfüllen: zum einen wollen wir für die kontinuierlichen Leser dieser Buchserie das Beispiel aus Band 3 entsprechend umschreiben, zum anderen für die Leser, die nur etwas über Simon's Basic wissen wollen, ein Beispiel bringen, in dem alle wichtigen Punkte zusammengefaßt sind.

Zunächst werden wir dabei auf das Definieren der Sprites eingehen und anschließend gleich auf ihre Bewegung. Für Spiele unerläßlich sind natürlich die Kollisionsprüfungen Sprite/Sprite und Sprite/Hintergrund. Auch das Vergrößern und Verkleinern von Sprites soll besprochen werden.

Da es doch ein recht komplexes Beispiel ist, was im folgenden Kapitel vorgestellt wird, haben wir gleich die Möglichkeit genutzt und andere Befehle von Simon's Basic in das Programm mit integriert, obwohl sie nicht direkt etwas mit Sprites zu tun haben.

Wer die Unterschiede in der Programmierung gegenüber anderen Methoden herausfinden möchte, der sei auf Anhang 1 verwiesen, wo das Komplettlisting des entsprechenden Beispiels aus Band 3 abgedruckt ist. Die Zeilennummern wurden in dem neuen Beispiel so vergeben, daß ein Vergleich relativ einfach möglich ist. Dies bedingt natürlich, daß die Zeilennummern nicht sauber vergeben werden konnten.

3.1 Sprites definieren

Da wir die Kenntnis der Behandlung von Sprites voraussetzen, wollen wir gleich mit mehrfarbigen Sprites beginnen. Die erste wichtige Zeile dafür in unserem Beispielprogramm ist

```
150 CMOB 14,0
```

Damit werden die beiden Farben eingeschaltet, die allen Sprites gemeinsam sind. Wir haben hier die Farben 14

(hellblau) und 0 (schwarz) gewählt. Für die Definition von Sprites muß zunächst der DESIGN-Befehl angewendet werden, der angibt, ob ein Sprite einfarbig (erster Parameter: 0) oder mehrfarbig (erster Parameter: 1) ist. Der zweite Parameter gibt die erste Speicherzelle des Blockes an, in dem die Daten des Sprites abgelegt werden sollen. In unserem Beispiel haben wir - üblicherweise - die Blocks 13, 14 und 15 herangezogen.

Dem DESIGN-Befehl folgen jeweils 21 Zeilen, denen ein 'Klammeraffe' vorangestellt ist, mit der Beschreibung der Sprites.

Hier zunächst unsere Beispiel-Sprites:

```
997 REM *******************************************
998 REM *   DATEN FUER SPRITE 0                   *
999 REM *******************************************
1000 DESIGN 1,13*64
1002 @.....BB.....
1004 @.....BB.....
1006 @.....BB.....
1008 @....CBBC....
1010 @....CBBC....
1012 @....CBBC....
1014 @....CBBC....
1016 @....CBBC....
1018 @.D..CBBC..D.
1020 @.D..CBBC..D.
1022 @.D.CCBBCC.D.
1024 @.DCCCBBCCCD.
1026 @.DCCCBBCCCD.
1028 @CDCCCBBCCCDC
1030 @CDCCCBBCCCDC
1032 @CDCCCBBCCCDC
1034 @CDCCCBBCCCDC
1036 @CDCCCBBCCCDC
1038 @...B.BB.B...
1040 @...B.BB.B...
1042 @...B.BB.B...
1097 REM *******************************************
1098 REM *   DATAS FUER SPRITE 1                   *
1099 REM *******************************************
1100 DESIGN 0,14*64
1102 @....BB............BB....
1104 @....BB............BB....
1106 @....BB............BB....
1108 @....BB............BB....
1110 @....BB............BB....
1112 @....BB............BB....
1114 @....BB............BB....
```

```
1116 @....BB............BB....
1118 @....BB............BB....
1120 @....BB............BB....
1122 @....BB............BB....
1124 @....BB............BB....
1126 @....BB............BB....
1128 @....BB............BB....
1130 @....BB............BB....
1132 @....BB............BB....
1134 @....BB............BB....
1136 @....BB............BB....
1138 @....BB............BB....
1140 @....BB............BB....
1142 @....BB............BB....
1197 REM ****************************************
1198 REM *   DATAS FUER SPRITE 2                *
1199 REM ****************************************
1200 DESIGN 0,15*64
1202 @BB....................BB
1204 @BB....................BB
1206 @BB....................BB
1208 @BB....................BB
1210 @BB....................BB
1212 @BB....................BB
1214 @BB....................BB
1216 @BB.........BB.........BB
1218 @BB........BBBB........BB
1220 @BB.......BBBBBB.......BB
1222 @BBBBBBBBBB....BBBBBBBBBB
1224 @BB.......BB..BB.......BB
1226 @BB........BBBB........BB
1228 @BB.........BB.........BB
1230 @BB....................BB
1232 @BB....................BB
1234 @BB....................BB
1236 @BB....................BB
1238 @BB....................BB
1240 @BB....................BB
1242 @BB....................BB
```

Wie man sieht, sind die Sprites 1 und 2 nur einfarbig dargestellt. In diesen Fällen wird durch den Buchstaben 'B' bestimmt, an welcher Stelle bei den Sprites ein farbiger Punkt erscheinen soll. Alle anderen Bildschirmpunkte werden durch einen einfachen '.' gekennzeichnet. Bei den mehrfarbigen Sprites (Sprite 0) werden die Buchstaben 'B', 'C' und 'D' zu Unterscheidung der einzelnen Farben herangezogen. Dabei steht das 'B' für die Farbe, die im CMOB-

Befehl als erstes angegeben ist, und 'D' für die zweite Farbe (hier: schwarz). Das 'C' ist Platzhalter für die dem Sprite eigene Farbe.

Wie man sieht, ist anhand des Listings das spätere Erscheinungsbild des Sprites offensichtlich. Die abstrakte Umdenkerei von DATA-Zeilen auf das spätere Aussehen - was sowieso kaum möglich war - entfällt. Ein Sprite-Editor kann entfallen, da diese direkt am Bildschirm (mittels Cursor) entwickelt werden können. Nach dem Zeichnen am Bildschirm sollten Sie natürlich nicht vergessen, die einzelnen Zeilen mit RETURN noch an das Programm zu übergeben.

Mit den bisher vorgestellten Befehlskombinationen werden die Daten für die Sprite beim Programmablauf in den entsprechenden Blocks generiert. Dabei ist es auch möglich, einer Spritenummer mehrere Bildinhalte zuzuordnen, wobei diese mit dem DESIGN-Befehl immer wieder in die entsprechenden Blocks geschrieben werden müssen, um sie an das Programm zu übergeben. Da dieser Befehl aber sehr schnell arbeitet, sind kaum Programmverzögerungen zu erwarten. An dem CMOB-Befehl und in Kapitel 1 haben wir schon gesehen, daß die Sprites bei Simon's Basic MOB (**M**oveble **O**bject **B**lock) genannt werden. Dieser Begriff ist auch in den meisten Sprite-Befehlen enthalten. So auch im nächsten Befehl, mit dem nach dem Aussehen der Sprites weitere Parameter festgelegt werden.

```
2000 MOB SET 0,13,7,0,1
2100 MOB SET 1,14,5,0,1
2200 MOB SET 2,15,1,0,0
```

Damit haben wir wieder einige übliche POKE-Befehle ersetzt. Vergleichen wir kurz mit dem Listing im Anhang. Dort werden in den Zeilen 2010, 2100 und 2200 die Blocks nochmals für die Spritenummern festgelegt. Das Abspeichern der Daten aus den DATA-Zeilen entfällt natürlich, dies übernimmt ja der DESIGN-Befehl. In dem Beispiel aus Band 3 werden die Sprites mit dem SPRITE-Befehl eingeschaltet. Ein GRENZEN-Kommando gibt es in Simon's Basic nicht. Das Festlegen der Farben in den Zeilen 2080, 2180 und 2280 wird auch von dem MOB SET-Befehl übernommen. Weiterhin legt dieser Befehl die Priorität gegenüber dem Hintergrund fest, die wir in dem anderen Beispiel nicht verwendet haben und gibt für das Register 28 den entsprechenden Wert, ob dieses Sprite in normaler Grafik oder Mehrfarbengrafik dargestellt werden soll (altes Beispiel: Zeile 2090). Wie wir an den drei Zeilen oben sehen, gibt der erste Parameter sicherlich die Spritenummer an und der zweite Parameter die jeweilige Blocknummer, die wir ja schon beim

DESIGN-Befehl kennengelernt haben. Wenn Sie das alte Beispiel bisher genau verfolgt haben und einen Vergleich mit dem MOB SET-Befehl durchgeführt haben, haben Sie sicherlich auch festgestellt, daß der dritte Parameter die dem Sprite eigene Farbe angibt. Der vierte Parameter gibt noch die Priorität gegenüber den Bildschirmtexten an und der fünfte Parameter die Farbauflösung. Die Werte im letzten Parameter (0,1) entsprechen wieder dem DESIGN-Befehl.

Daß manche Angaben doppelt erfolgen, liegt an der beibehaltenen Struktur des Commodore 64. Ein als mehrfarbiges Sprite definierter MOB muß nicht unbedingt bei seiner Darstellung mehrfarbig angesprochen werden. Diesen Effekt haben wir ja auch bereits schon in Band 3 besprochen.

Nun haben wir zwar schon die Sprites entworfen und die Parameter definiert, aber auf dem Bildschirm wird noch nichts dargestellt. Dazu dient der MMOB-Befehl. Damit wird das Sprite am Bildschirm positioniert und kann auch bewegt werden, seine Größe wird festgelegt und die Geschwindigkeit für die Bewegung. Da wir in unserem Beispiel Sprite 1 erst auf Anweisung des Anwenders sichtbar machen wollen, werden im Vorspann nur Sprites 0 und Sprites 2 vorbesetzt mit:

```
2030 MMOB 0,X0,Y0,X0,Y0,0,255
2230 MMOB 2,X2,Y2,X2,Y2,0,255
```

Nachdem wir vorher definiert haben

```
2010 X0=30
2020 Y0=50
2210 X2=100
2220 Y2=100
```

Aus mnemotechnischen Gründen wollen wir die Variablen X0 und Y0 für die X- und Y-Koordinaten von Sprite 0 verwenden, X1 und Y1 entsprechend für Sprite 1, und X2 und Y2 für Sprite 2.

Wie in allen Sprite-Befehlen gibt der erste Parameter auch wieder die Nummer des Sprites an, die von 0 bis 7 durchnumeriert sind. Im MMOB-Bfehl gibt das erste Koordinatenpaar den Punkt an, an dem das Sprite am Bildschirm auftaucht (Berechnung im Normalgrafikmodus) und - falls es bewegt werden soll - praktisch den Ausgangspunkt darstellt. Das zweite Koordinatenpaar (4. und 5. Parameter) stellen die Zielkoordinaten dar. Da wir unsere Sprites zu Beginn nicht bewegen wollen, haben wir beide Koordinatenpaare gleich gewählt.

Der vorletzte (6. Parameter) gibt die Größe des Sprites an, wobei wir alle Sprite zunächst in Normalgröße darstellen wollen (vergleiche Kapitel 3.5).

Der letzte Parameter gibt die Geschwindigkeit an, die von 1 bis 255 reguliert werden kann, wobei (unlogischerweise) 255 die langsamste Geschwindigkeit darstellt. Achtung: Im Gegensatz zu den in Band 3 vorgestellten Bewegunsroutinen für Sprites werden bei Simon's Basic erst die Bewegungen durchgeführt, bevor das Basic-Programm fortgesetzt wird. Eine Interruptsteuerung liegt nicht vor.

Damit haben wir unsere Sprites entworfen, die Parameter festgelegt und sie auf dem Bildschirm dargestellt. Im nächsten Kapitel wollen wir auf die Bewegung der Sprites eingehen.

3.2 Sprites bewegen

Es gibt sicherlich andere Möglichkeiten und Aufgabenstellungen Sprites zu bewegen, als die in diesem Kapitel vorgestellte. Wenn Sie die Sprites (über Tastatur oder auch Joystick) steuern wollen, so kann Ihnen das vorliegende Beispiel jedoch als Grundlage dienen. Andere Bewegungsformen (hüpfende Känguruhs) sind im Handbuch dargestellt.

Auch in diesem Beispiel wollen wir Geschwindigkeitsänderungen zulassen. Die Bewegung wird über den RLOCMOB-Befehl durchgeführt. Mit diesem Befehl werden zunächst die Spritenummer, dann Zielkoordinaten und schließlich noch (wie bei dem MMOB-Befehl) die Größe und Geschwindigkeit des Sprites festgelegt.

Beschränken wir uns zunächst wieder auf die Bewegung von Sprite 0. Zunächst wollen wir wieder den Rahmen für die Bewegung schaffen mit

```
3000 PRINT"(CLR/HOME)"
3002 GETA$
3004 IF A$ NE "" THEN C$=B$ : B$=A$    (NE für ungleich)
```

Wie in den anderen Bänden, soll auch hier die Steuerung des Sprites über neun Buchstabentasten wie folgt erfolgen:

Q - oben links
W - oben
E - oben rechts
D - rechts
C - rechts unten

X - unten
Z - unten links
A - links
S - stop / anhalten

Wenn das Sprite anhalten soll, (S) so sind die Zielkoordinaten gleich den Startkoordinaten und wir können folgenden Befehl eingeben:

```
3030 IF B$="S" THEN : RLOCMOB 0,X0,Y0,E0,255 : GOTO3002
```

Wie schon gesagt, gibt beim RLOCMOB-Befehl der erste Parameter die Spritenummer an, die beiden nächsten die Zielkoordinaten und die beiden letzten Größe und Geschwindigkeit. Da die Zielkoordinaten in diesem Fall gleich den Anfangskoordinaten sind, kann man die Geschwindigkeit frei wählen. Ansonsten wollen wir für Größe und Geschwindigkeit folgende Variable verwenden:

E0 - Größe Sprite 0
E1 - Größe Sprite 1
E2 - Größe Sprite 2 (vergl. auch Kapitel 3.5)
G0 - Geschwindigkeit Sprite 0
G1 - Geschwindigkeit Sprite 1
G2 - Geschwindigkeit Sprite 2

Den Sprung auf Zeile 3002 geben wir aus Rechenzeitgründen vor. Da wir Fehlermeldungen von Simon's Basic vermeiden wollen wenn wir einen Rand erreichen, müssen wir vor einer Bewegung des Sprites noch die Koordinaten überprüfen mit

```
3032 IF X0 LT 1 THEN X0=1
3034 IF X0 GT 318 THEN X0=318
3036 IF Y0 LT 1 THEN Y0=1
3038 IF Y0 GT 198 THEN Y0=198
```

Da der verwendete Typenraddrucker das 'Kleinerzeichen' und 'Größerzeichen' nicht darstellen kann (siehe auch Parametertypen im Anhang) werden sie durch die in FORTRAN üblichen Bezeichnungen LT und GT ersetzt.

In den folgenden Zeilen werden jeweils die Sprites in die gewünschte Richtung bewegt und anschließend die Koordinaten des Sprite mit den Werten für die Zielkoordinaten besetzt. Dazu wird jeweils der Wert 'G0' addiert oder subtrahiert. In G0 wird also die Geschwindigkeit in Punkten pro Bewegung angegeben.

```
3040 IF B$="D" THEN : RLOCMOB 0,X0+G0,Y0,E0,G : X0=X0+G0
                    : GOTO3002
3050 IF B$="E" THEN : RLOCMOB 0,X0+G0,Y0-G0,E0,G :
```

```
                        X0=X0+G0 : Y0=Y0-G0 : GOTO3002
3060 IF B$="W" THEN : RLOCMOB 0,X0,Y0-G0,E0,G :
                        Y0=Y0-G0 : GOTO 3002
3070 IF B$="Q" THEN : RLOCMOB 0,X0-G0,Y0-G0,E0,G :
                        X0=X0-G0 :Y0=Y0-G0 : GOTO3002
3080 IF B$="A" THEN : RLOCMOB 0,X0-G0,Y0,E0,G :
                        X0=X0-G0 : GOTO 3002
3090 IF B$="Z" THEN : RLOCMOB 0,X0-G0,Y0+G0,E0,G :
                        X0=X0-G0 : Y0=Y0+G0 : GOTO3002
3100 IF B$="X" THEN : RLOCMOB 0,X0,Y0+G0,E0,G : Y0=Y0+G0 :
                        GOTO 3002
3110 IF B$="C" THEN : RLOCMOB 0,X0+G0,Y0+G0,E0,G :
                        X0=X0+G0 : Y0=Y0+G0 : GOTO3002
```

Damit wäre die Bewegung für Sprite 0 abgeschlossen und wir können die Bewegung für Sprite 1 besprechen. Sprite 1 stellt den 'Schuß' dar und soll eingeschaltet werden, wenn die Taste mit dem Pfeil nach oben gedrückt wird. Die Koordinaten beim Abschuß sind natürlich die von Sprite 0 und entsprechend werden die Variablen X1 und Y1 auch besetzt. Der Schuß soll jeweils in unserem Beispiel am Bildschirm nur nach oben gehen, so daß nur die Y-Koordinate um eins zu vemindern ist. Außerdem muß noch ein Merker gesetzt werden, daß bereits das Sprite mit dem Schuß aktiviert wurde, und für die Bewegung von Sprite 0 wird noch der alte Wert (C$) herangezogen. Die Geschwindigkeit wurde absichtlich auf '1' gesetzt, damit auch eine korrekte Kollisionsprüfung stattfinden kann. Natürlich ist dieses Beispiel kein atemberaubendes Spiel, da das Basic die Sache - trotz der Hilfsmittel - recht langsam ablaufen läßt. Der erste Befehl für unseren Schuß könnte also lauten:

```
3120 IFB$="(Pfeil nach oben)" THEN X1=X0 : Y1=Y0 :
      MMOB 1,X1,Y1,X1,Y1-1,E1,1 : S=1 : B$=C$ : GOTO3002
```

Da wir die Abgabe eines Schusses sperren wollen, wenn bereits Sprite 1 eingeschaltet ist, fügen wir noch ein

```
3115 IF S THEN 3135
```

Damit Sprite 1 auch im folgenden weiter bewegt wird, fügen wir in das Bewegungsmenü für Sprite 0 noch ein

```
3014 IF S THEN Y1=Y1-G0 : RLOCMOB 1,X1,Y1,E1,1 :
        IF Y0 LT 5 THEN S=0 : MOB OFF 1
```

wobei zunächst die entsprechende Y-Koordinate um die Geschwindigkeit von Sprite 0 vermindert wird - damit bewegen sich beide mit gleicher Geschwindigkeit. Das Sprite wird mit dem RLOCMOB-Befehl bewegt und ausgeschaltet, wenn die Y-Koordinate kleiner als 5 ist. Beim Ausschalten des Spri-

tes muß natürlich auch der Merker zurückgesetzt werden. Das eigentliche Ausschalten wird mit dem Befehl MOB OFF durchgeführt. Außer der Spritenummer beinhaltet dieser Befehl keinen weiteren Parameter.

Da wir keinen automatischen Befehl zum Reflektieren an den Grenzen zur Verfügung haben, werden wir Sprite 2 wieder zufällig bewegen (vergleiche Band 1), wobei wir für jede Richtung eine eigene Zufallszahl zwischen -5 und 5 festlegen wollen. Diese Werte müssen natürlich noch auf Plausibiltät geprüft werden. Anhand dieser Werte können wir anschließend das Sprite bewegen und das Programmstück sieht dann wie folgt aus:

```
6000 ZX=INT(10*RND(6))-5
6010 ZY=INT(10*RND(7))-5
6020 X2=X2+ZX
6030 Y2=Y2+ZY
6040 IF X2<10 THEN X2=10
6050 IF X2>300 THEN X2=300
6060 IF Y2<10 THEN Y2=10
6070 IF Y2>195 THEN Y2=195
6080 RLOCMOB 2,X2,Y2,E2,1
6090 RETURN
```

Damit das Unterprogramm auch aufgerufen wird, ergänzen wir

```
3006 GOSUB 6000
```

Vergleichen wir das ganze nun mit unserem Beispiel aus Band 3. Zunächst stellen Sie - rein optisch - fest, daß die Zeilen im Bewegungsmenü für Sprite 0 länger sind. Dies kommt durch unterschiedliche Konzeption des Bewegungsbefehls. Der FAHRT-Befehl benötigt keine Zielkoordinaten, sondern nur die Richtung und die Geschwindigkeit. Da das Sprite solange bewegt wird bis es ein anderes FAHRT-Kommando erhält, ist natürlich das Nachrechnen der jeweiligen Position unnötig. Dies fällt besonders bei der Bewegung von Sprite 1 auf, da dies in dem Unterprogramm ab Zeile 5000 nur einmal gestartet zu werden braucht, und sonst keine weitere Handhabung erforderlich ist. Das Starten ist natürlich hinreichend kompliziert, da die aktuelle Position von Sprite 0 im Moment des Starts festgestellt werden muß (5010 und 5020).

Noch einfacher ist die Bewegung von Sprite 2, da dies über den GRENZEN-Befehl die obere Hälfte des Bildschirmbereiches für die Bewegung zugewiesen bekommen hat, wobei alle Seiten dieses Sprites reflektierend sind. Damit entfällt jede weitere Behandlung für Sprite 2 im alten Beispiel.

Geschwindigkeit ändern

Da wir in dem MMOB-Befehl und dem RLOCMOB-Befehl jeweils erneut die Geschwindigkeit angeben müssen, haben wir die Variablen G0, G1 und G2 zur Geschwindigkeitsbestimmung herangezogen. In einem kleinen Unterprogramm wollen wir nun die Möglichkeit der Änderung der Geschwindigkeit für Sprite 0 zulassen. Zuvor müssen wir jedoch noch - wie im alten Beispiel auch - aufgrund des Tastendrucks 'G' in ein Unterprogramm verzweigen mit

```
3135 IF B$="G" THEN GOSUB 3500 : GOTO 3002
```

In dem Unterprogramm soll keine Bildschirmabfrage erwünscht sein, so daß wir über den GET-Befehl die Tastatur abfragen müssen. Die Geschwindigkeit '0' wollen wir nicht zulassen, da diese auch über den STOP-Befehl ('S') erreicht werden kann. Da über die Tastatur somit die Geschwindigkeiten 1 bis 9 zur Verfügung stehen, wollen wir den erhaltenen Wert noch mit 2 multiplizieren um die Schrittweite für die Bewegungsbefehle um jeweils zwei Bildschirmpunkte zu erhöhen. Auch muß die alte Bewegungsrichtung wieder an die Variable für die Bewegungsverteilung von Sprite 0 (B$) übergeben werden und unser Unterprogramm kann dann wie folgt aussehen:

```
3497 REM ********************************************
3498 REM *   GESCHWINDIGKEIT AENDERN                *
3499 REM ********************************************
3500 GETG$:IFG$=""THEN3500
3510 G=VAL(G$)
3520 IFG<1THEN3500
3530 G0=G*2
3540 B$=C$
3550 RETURN
```

3.3 Sprite/Sprite-Kollision

Auch die Kollisionsprüfung wollen wir in einem Unterprogramm durchführen und ergänzen die Zeile 3006 wie folgt

```
3006 GOSUB 6000 : GOSUB 7000
```

Wenn Sprite 2 getroffen ist, wollen wir das Programm mit

dem RUN-Befehl einfach neu starten, um das Rücksetzen aller Parameter zu vermeiden. Dies ist ja auch nicht weiter schlimm, da sowieso keine Punkte oder ähnliches in der vorliegenden Version vergeben werden.

Zunächst müssen wir die Kollisionsabfrage vorbereiten, indem wir zwei DETECT-Befehle vergeben: den einen zum Löschen des Kollisionsregisters und dem anderen zum Abfragen der entsprechenden Bits in diesem Register. Durch diese Behandlung bedingt, wird natürlich nicht immer sofort festgestellt, ob sich zwei Sprites berühren, was Sie am Bildschirm auch beobachten können. D.h. die manchmal unterbleibende Feststellung einer Kollision vom Betriebssystem her, tritt nun noch im verstärktem Maße auf.

Da wir zwei verschiedene Behandlungen machen wollen, wenn entweder das Geschoß oder unser 'Raumgleiter' mit Sprite 2 kollidieren, führen wir beim CHECK-Befehl noch eine entsprechende Abfrage durch, und unser Programmstück hat somit folgendes Aussehen.

```
7000 DETECT 0
7010 DETECT 0
7020 IF CHECK(1,2)=0 THEN GOTO 7200
7030 IF CHECK(0,2)=0 THEN GOTO 7100
7040 RETURN
```

Behandeln wir zunächst den Fall, daß Sprite 1 und 2 kollidiert sind. Zunächst schalten wir Sprite 2 mit MOB OFF ab. An dieser Stelle möchten wir eine Neuerung gegenüber der alten Version bringen, wo wir einfach die Farben des Sprites geändert haben. Damit wollen wir auch aufzeigen, daß mit dem DESIGN-Befehl auch eine Umschaltung von Sprites möglich ist. In dem frei gewordenen 15. Block nehmen wir neue Daten auf, die ein explodierendes Raumschiff darstellen sollen (sicherlich können Sie dies noch etwas besser). Sprite 1 kann dann ausgeschaltet werden und auch Sprite 0 (wenn dies nicht an der Kollision beteiligt war). Als Abschluß wollen wir mit unterschiedlichen Pausen noch alle drei Farben von Sprite 2 verändern. Für die den Sprites allgemein zugedachten Farben ist der CMOB-Befehl zuständig und für die spriteeigene Farbe müssen wir jeweils den Befehl MOB SET benutzen. Als Programm:

```
7200 MOB OFF 2
7201 DESIGN 1,15*64
7202 @...DCBBB....
7204 @..DCBDCBCB..
7206 @...DCBCBDB..
7208 @.DCBDCBBBBDC
```

```
7210 @DCBCDCBDCBCB
7212 @DBBCBCBCBCDC
7214 @DDBBBCDCDDCD
7216 @DCBDCBDCBCBC
7218 @BCDCBCBDCBCB
7220 @DBDBDBDCBCBC
7222 @DCBDBDCBDCBC
7224 @DCBBBCBCB...
7226 @DBDBCBCB....
7228 @DBBBBCDCB...
7230 @DBCBCDCBDB..
7232 @DCBCBDCB....
7234 @....DCBBBBCD
7236 @...DCBCBDCB.
7238 @..DBBBDCBBD.
7240 @.DBBDBDBBDD.
7242 @...DCBDCB...
7244 MOB SET 2,15,7,0,1
7246 MOB OFF 1
7247 IF T0=0 THEN MOB OFF 0
7248 FOR I=1 TO 20
7250 CMOB 0,8
7251 PAUSE 0.9
7252 MOB SET 2,15,8,0,1
7253 PAUSE 0.5
7254 CMOB 7,8
7255 PAUSE 0.1
7256 MOB SET 2,15,7,0,1
7257 PAUSE 0.8
7258 CMOB 0,1
7259 PAUSE 0.7
7260 MOB SET 2,15,0,0,1
7261 PAUSE 0.5
7262 CMOB 7,8
7263 PAUSE 0.2
7264 MOB SET 2,15,1,0,1
7265 PAUSE 0.4
7266 CMOB 1,6
7267 PAUSE 0.1
7268 NEXT
7270 RUN
```

Bei einer Kollision zwischen den beiden Raumschiffen wollen wir dasselbe durchführen, wobei auch das zweite Raumschiff explodieren soll. Da die beiden Sprites sowieso zwei Farben gemeinsam haben, genügt es einfach, wenn ein neuer MOB SET-Befehl gegeben wird, der auch auf das eben definierte Sprite zurückgreift. Hiermit ist auch ein Beispiel gegeben, wie zwei verschiedene Sprites auf die gleiche Definition zurückgreifen.

Damit Sprite 0 in Zeile 7247 nicht abgeschaltet wird, müssen wir noch den Merker T0 auf '1' setzen und erhalten somit

```
7100 MOB SET 0,15,6,0,1
7110 T0=1
```

Der Übergang zu Zeile 7200 braucht nicht programmiert werden, da diese sowieso folgt. Wenn Sie das Programmstück ab Zeile 7100 ändern, sollten Sie darauf achten und entsprechend ein 'GOTO 7200' einfügen.

3.4 Sprite/Hintergrund-Kollision

Auch im vorliegenden Beispiel wollen wir die Sprite/Hintergrund-Kollision an der 'Landebasis' durchführen. Durch andere Befehle von Simon's Basic (Bildschirm rollen) erhalten wir eine weitere Vereinfachung. Zunächst können wir das Feld H$() mit der Landebasis vergessen und die Landebasis direkt am Bildschirm ausgeben. Dem alleinigen Einschalten von Sprite 1 in Zeile 8100 (altes Beispiel) entspricht das Abschalten von Sprite 1 und 2. Das Positionieren unserer Rakete im Ausgangszustand erledigen wir wieder mit dem RLOCMOB-Befehl.

Um die Basis ins Bild zu schieben, benutzen wir den Befehl RIGHTW für den ganzen Bildschirm. Da sich außer Landebasis nur Sprites auf dem Bildschirm befinden und diese nicht mitgerollt werden, ist es egal, ob wir zyklisch rollen oder mit Nachziehen von Leerzeilen bzw. -spalten. Genau so gut könnten wir den Befehl RIGHTB benutzen.

Mit dem Rücksetzen der Bewegungsrichtung für Sprite 0 und dem Setzen eines Merkers können wir das Unterprogramm beenden und erhalten:

```
8000 REM ****************************************
8010 REM *   LANDUNG                            *
8020 REM ****************************************
8030 PRINT"                                     "
8040 PRINT"              "
8050 PRINT"              "
8060 PRINT"             "
8070 PRINT"            "
8080 PRINT"           "
8090 PRINT"           "
8100 MOB OFF 1 : MOB OFF 2
8110 REM ----- RAKETE IN AUSGANGSZUSTAND BRINGEN -
```

```
8120 X0=150
8130 Y0=70
8135 RLOCMOB 0,X0,Y0,0,50
8140 REM ----- BASIS IN BILD SCHIEBEN -----------
8150 FORI=1TO25
8160 RIGHTW 0,0,39,24
8220 REM ----- LAENGER WERDENDE WARTESCHLEIFE ----
8230 FOR J=1 TO 10*I
8240 NEXT
8250 NEXT
8260 REM ----- ANFLUGVORBEREITUNG ---------------
8270 B$=C$
8280 L=1
8300 RETURN
```

Um nun festzustellen, ob die Landung geglückt ist oder nicht, fügen wir im Hauptprogramm noch ein:

```
3008 DETECT 1 : DETECT 1
3010 IF L THEN IF CHECK(0)=0 THEN GOTO 8540
3012 IF L AND X0 GT 238 AND X0 LT 242 AND Y0 GT 180 THEN
         GOSUB9030
```

Damit eine Bewegung von Sprite 1 und Sprite 2 nicht unnötig durchgeführt wird, wenn eine Landung vorgenommen wird, ändern wir Zeile 3006 noch ab wie folgt:

```
3006 IF NOT L THEN GOSUB6000 : GOSUB 7000
```

Damit haben wir erreicht, daß die Programmsequenzen zum Bewegen von Sprite 0 weiterhin genutzt werden kann.

In Zeile 3008 wird zunächst das Register für die Sprite/Hintergrund-Kollision gelöscht und mit dem zweiten DETECT-Befehl werden die Werte in das Register eingetragen. In Zeile 3010 wird die Hintergrundkollision nur abgefragt, wenn die Variabel L ungleich 0 ist. Übrigens spart die Anweisung IF...THEN IF... gegenüber IF...AND...THEN Rechenzeit.

Bei der mißglückten Landung wollen wir auf unser Programmstück ab Zeile 7200 zurückgreifen und die Landebasis mit dem FLASH-Befehl blinken lassen. Damit erhalten wir:

```
8550 FLASH 8,10
8560 RLOCMOB 2,X0,Y0,0,1
8570 GOTO7200
```

Da wir das ganze noch etwas besser gestalten wollen und außerdem den FLASH-Befehl rückgängig machen müssen schreiben wir das Programmstück ab Zeile 7200 wie folgt um:

```
7270 MOB OFF 2
7272 OFF
7274 PAUSE 5
7276 RUN
```

Bei der erfolgreichen Landung brauchen wir fast gar nichts zu ändern. Da eine Bewegung zu diesem Zeitpunkt nicht mehr abgefragt wird, kann der FAHRT-Befehl in Zeile 9040 (altes Beispiel) entfallen. Ebenso Zeile 9090 da die Variable H1$ nicht mehr gebraucht wird. Statt des Einschaltens von Sprite 0 und 2 in Zeile 9080 muß nun noch Sprite 2 mit dem MOB SET-Befehl neu gesetzt werden und wir erhalten folgendes Programmstück:

```
9000 REM *****************************************
9010 REM *   BEHANDLUNG ERFOLGREICHE LANDUNG     *
9020 REM *****************************************
9030 L=0
9050 PRINT"SQQQQQQQQQQQ      ]LANDUNG ERFOLGREICH
9060 FORI=1TO2000
9070 NEXT
9080 MOB SET 2,15,1,0,0
9100 PRINT"[
9110 RETURN
```

3.5 Sprite vergrößern und verkleinern

Nachdem wir das Definieren von Sprites, ihre Bewegung und mögliche Kollision behandelt haben, bleibt nur noch die Vergrößerung bzw. Verkleinerung von Sprites.

Da die Größe von Sprites beim MMOB-Befehl und beim RLOC-MOB-Befehl mit behandelt wird, brauchen wir nun noch die entsprechenden Variablen (E0, E1 u. E2) zu besetzen. Da sowohl X als auch Y-Richtung in einem behandelt werden können, brauchen wir im Hauptprogramm nur noch abzufragen, ob die Taste 'V' für Vergrößern oder Verkleinern gewählt wurde und schreiben

```
3138 IF B$="V" THEN GOSUB 9500
```

In den Variablen für Vergrößerung und Verkleinerung können jeweils die Werte 0 bis 3 stehen, wobei bedeuten:

0 - keine Vergrößerung
1 - Vergrößerung in X-Richtung
2 - Vergrößerung in Y-Richtung
3 - Vergrößerung in beiden Richtungen

Zunächst wollen wir die Vergrößerung abfragen und dann die entsprechende Spritenummer. Aufgrund der Spritenummer wollen wir in eine Zeile verzweigen, wo die entsprechende Variable auf die vorher erfaßte Vergrößerung gesetzt wird.

```
9500 B$=C$
9510 GETA$
9520 IFA$=""THEN9510
9530 A=VAL(A$)
9540 IFA>3THEN9510
9550 GETNR$
9560 IFNR$=""THEN9550
9570 NR=VAL(NR$)
9580 IFNR>2THEN9550
9590 ON NR+1 GOTO 9600,9610,9620
9600 E0=A : RETURN
9610 E1=A : RETURN
9620 E2=A : RETURN
```

Aber die Vergrößerung wollen wir auch noch an einer anderen Stelle verwenden, nämlich bei der 'Explosion' der Sprites und fügen deshalb ein

```
7269 GOSUB 7300
```

Dann kopieren wir einen Teil der Zeilen ab 7248. Vorher müssen wir jedoch noch die Größe des Sprites verändern und erhalten somit:

```
7300 FOR I=1 TO 20
7301 RLOCMOB 2,X2,Y2,3,1
7302 MOB SET 2,15,8,0,1
7303 PAUSE 0.5
7304 CMOB 7,8
7305 PAUSE 0.1
7306 MOB SET 2,15,7,0,1
7307 PAUSE 0.8
7308 CMOB 0,1
7309 PAUSE 0.7
7310 MOB SET 2,15,0,0,1
7311 PAUSE 0.5
7312 CMOB 7,8
7313 PAUSE 0.2
7314 MOB SET 2,15,1,0,1
7315 NEXT
7316 RETURN
```

```
100 REM ****************************************
101 REM *      R A K E T E / S P R I T E S     *
102 REM ****************************************
110 COLOUR 6,6
150 CMOB 14,0
997 REM ****************************************
998 REM *   DATEN FUER SPRITE 0                *
999 REM ****************************************
1000 DESIGN 1,13*64
1002 @.....BB.....
1004 @.....BB.....
1006 @.....BB.....
1008 @....CBBC....
1010 @....CBBC....
1012 @....CBBC....
1014 @....CBBC....
1016 @....CBBC....
1018 @.D..CBBC..D.
1020 @.D..CBBC..D.
1022 @.D.CCBBCC.D.
1024 @.DCCCBBCCCD.
1026 @.DCCCBBCCCD.
1028 @CDCCCBBCCCDC
1030 @CDCCCBBCCCDC
1032 @CDCCCBBCCCDC
1034 @CDCCCBBCCCDC
1036 @CDCCCBBCCCDC
1038 @...B.BB.B...
1040 @...B.BB.B...
1042 @...B.BB.B...
1097 REM ****************************************
1098 REM *   DATAS FUER SPRITE 1                *
1099 REM ****************************************
1100 DESIGN 0,14*64
1102 @....BB............BB....
1104 @....BB............BB....
1106 @....BB............BB....
1108 @....BB............BB....
1110 @....BB............BB....
1112 @....BB............BB....
1114 @....BB............BB....
1116 @....BB............BB....
1118 @....BB............BB....
1120 @....BB............BB....
1122 @....BB............BB....
1124 @....BB............BB....
1126 @....BB............BB....
1128 @....BB............BB....
1130 @....BB............BB....
1132 @....BB............BB....
1134 @....BB............BB....
```

```
1136 @....BB............BB....
1138 @....BB............BB....
1140 @....BB............BB....
1142 @....BB............BB....
1197 REM *****************************************
1198 REM *   DATAS FUER SPRITE 2                 *
1199 REM *****************************************
1200 DESIGN 0,15*64
1202 @BB....................BB
1204 @BB....................BB
1206 @BB....................BB
1208 @BB....................BB
1210 @BB....................BB
1212 @BB....................BB
1214 @BB....................BB
1216 @BB.........BB.........BB
1218 @BB........BBBB........BB
1220 @BB.......BBBBBB.......BB
1222 @BBBBBBBBBB....BBBBBBBBBB
1224 @BB.......BB..BB.......BB
1226 @BB........BBBB........BB
1228 @BB.........BB.........BB
1230 @BB....................BB
1232 @BB....................BB
1234 @BB....................BB
1236 @BB....................BB
1238 @BB....................BB
1240 @BB....................BB
1242 @BB....................BB
1997 REM *****************************************
1998 REM *   SPRITE 0 VORBESETZEN                *
1999 REM *****************************************
2000 MOB SET 0,13,7,0,1
2010 X0=30
2020 Y0=50
2030 MMOB 0,X0,Y0,X0,Y0,E0,255
2097 REM *****************************************
2098 REM *   SPRITE 1 VORBESETZEN                *
2099 REM *****************************************
2100 MOB SET 1,14,5,0,1
2110 MOB OFF 1
2197 REM *****************************************
2198 REM *   SPRITE 2 VORBESETZEN                *
2199 REM *****************************************
2200 MOB SET 2,15,1,0,0
2210 X2=100
2220 Y2=100
2230 MMOB 2,X2,Y2,X2,Y2,E2,255
2997 REM *****************************************
2998 REM *   BEWEGUNGSABLAUF                     *
2999 REM *****************************************
```

```
3000 PRINT"♥"
3001 G0=2
3002 GETA$
3004 IF A$<>"" THEN C$=B$ : B$=A$
3006 IF NOT L THEN GOSUB6000 : GOSUB 7000
3008 DETECT 1 : DETECT 1
3010 IF L THEN IF CHECK(0) =0 THEN GOTO 3540
3012 IF L AND X0>238 AND X0<242 AND Y0>180 THEN
                                         GOSUB9030
3014 IF S THEN Y1=Y1-G0 : RLOCMOB 1,X1,Y1,E1,1
                    : IFY1<5THENS=0:MOB OFF 1
3030 IF B$="S"THEN : RLOCMOB 0,X0,Y0,E0,255 : GOTO3002
3032 IF X0<1 THEN X0=1
3034 IF X0>318 THEN X0=318
3036 IF Y0<1 THEN Y0=1
3038 IF Y0>198 THEN Y0=198
3040 IF B$="D" THEN : RLOCMOB 0,X0+G0,Y0,E0,G : X0=X0+G0
                                                : GOTO3002
3050 IFB$="E"THEN:RLOCMOB 0,X0+G0,Y0-G0,E0,G:X0=X0+G0
                                    :Y0=Y0-G0:GOTO3002
3060 IFB$="W"THEN:RLOCMOB 0,X0,Y0-G0,E0,G:Y0=Y0-G0
                                           :GOTO3002
3070 IFB$="Q"THEN:RLOCMOB 0,X0-G0,Y0-G0,E0,G:X0=X0-G0
                                    :Y0=Y0-G0:GOTO3002
3080 IFB$="A"THEN:RLOCMOB 0,X0-G0,Y0,E0,G:X0=X0-G0
                                           :GOTO3002
3090 IFB$="Z"THEN:RLOCMOB 0,X0-G0,Y0+G0,E0,G:X0=X0-G0
                                    :Y0=Y0+G0:GOTO3002
3100 IFB$="X"THEN:RLOCMOB 0,X0,Y0+G0,E0,G:Y0=Y0+G0
                                           :GOTO3002
3110 IFB$="C"THEN:RLOCMOB 0,X0+G0,Y0+G0,E0,G:X0=X0+G0
                                    :Y0=Y0+G0:GOTO3002
3115 IFSTHEN3135
3120 IFB$="↑"THENX1=X0:Y1=Y0:MMOB 1,X1,Y1,X1,Y1-1,E1,1
                                    :S=1:B$=C$:GOTO3002
3121 IFB$="↑"THENB$=C$:GOTO3002
3135 IFB$="G"THENGOSUB3500:GOTO3002
3136 IFB$="L"THENGOSUB8030
3138 IFB$="V"THENGOSUB9500
3140 GOTO3002
3497 REM ****************************************
3498 REM *   GESCHWINDIGKEIT AENDERN           *
3499 REM ****************************************
3500 GETG$:IFG$=""THEN3500
3510 G=VAL(G$)
3520 IFG<1THEN3500
3530 G0=G*2
3540 B$=C$
3550 RETURN
5997 REM ****************************************
5998 REM *   SPRITE 2 BEWEGEN                  *
5999 REM ****************************************
```

```
6000 ZX=INT(10*RND(6))-5
6010 ZY=INT(10*RND(7))-5
6020 X2=X2+ZX
6030 Y2=Y2+ZY
6040 IF X2<10 THEN X2=10
6050 IF X2>300 THEN X2=300
6060 IF Y2<10 THEN Y2=10
6070 IF Y2>195 THEN Y2=195
6080 RLOCMOB 2,X2,Y2,E2,1
6090 RETURN
6997 REM ****************************************
6998 REM *   KOLLISIONSPRUEFUNG                 *
6999 REM ****************************************
7000 DETECT 0
7010 DETECT 0
7020 IF CHECK(1,2)=0 THEN GOTO 7200
7030 IF CHECK(0,2)=0 THEN GOTO 7100
7040 RETURN
7100 MOB SET 0,15,6,0,1
7110 T0=1
7197 REM ****************************************
7198 REM *   SPRITE 2 GETROFFEN                 *
7199 REM ****************************************
7200 MOB OFF 2
7201 DESIGN 1,15*64
7202 @...DCBBB....
7204 @..DCBDCBCB..
7206 @...DCBCBDB..
7208 @.DCBDCBBBBDC
7210 @DCBCDCBDCBCB
7212 @DBBCBCBCBCDC
7214 @DDBBBCDCDDCD
7216 @DCBDCBDCBCBC
7218 @BCDCBCBDCBCB
7220 @DBDBDBDCBCBC
7222 @DCBDBDCBDCBC
7224 @DCBBBCBCB...
7226 @DBDBCBCB....
7228 @DBBBBCDCB...
7230 @DBCBCDCBDB..
7232 @DCBCBDCB....
7234 @....DCBBBBCD
7236 @...DCBCBDCB.
7238 @..DBBBDCBBD.
7240 @.DBBDBDBBDD.
7242 @...DCBDCB...
7244 MOB SET 2,15,7,0,1
7246 MOB OFF 1
7247 IF T0=0 THEN MOB OFF 0
7248 FOR I=1 TO 20
7250 CMOB 0,8
```

```
7251 PAUSE 0.9
7252 MOB SET 2,15,8,0,1
7253 PAUSE 0.5
7254 CMOB 7,8
7255 PAUSE 0.1
7256 MOB SET 2,15,7,0,1
7257 PAUSE 0.8
7258 CMOB 0,1
7259 PAUSE 0.7
7260 MOB SET 2,15,0,0,1
7261 PAUSE 0.5
7262 CMOB 7,8
7263 PAUSE 0.2
7264 MOB SET 2,15,1,0,1
7265 PAUSE 0.4
7266 CMOB 1,6
7267 PAUSE 0.1
7268 NEXT
7269 GOSUB 7300
7270 MOB OFF 2
7272 OFF
7274 PAUSE 5
7276 RUN
7300 FOR I=1 TO 20
7301 RLOCMOB 2,X2,Y2,3,1
7302 MOB SET 2,15,8,0,1
7303 PAUSE 0.5
7304 CMOB 7,8
7305 PAUSE 0.1
7306 MOB SET 2,15,7,0,1
7307 PAUSE 0.8
7308 CMOB 0,1
7309 PAUSE 0.7
7310 MOB SET 2,15,0,0,1
7311 PAUSE 0.5
7312 CMOB 7,8
7313 PAUSE 0.2
7314 MOB SET 2,15,1,0,1
7315 NEXT
7316 RETURN
8000 REM ****************************************
8010 REM *   LANDUNG                            *
8020 REM ****************************************
8030 PRINT"                                 "
8040 PRINT"              "
8050 PRINT"              "
8060 PRINT"              "
8070 PRINT"              "
8080 PRINT"              "
8090 PRINT"              "
8100 MOB OFF 1 : MOB OFF 2
```

```
8110 REM ----- RAKETE IN AUSGANGSZUSTAND BRINGEN -
8120 X0=150
8130 Y0=70
8135 RLOCMOB 0,X0,Y0,0,50
8140 REM ----- BASIS IN BILD SCHIEBEN -----------
8150 FORI=1TO25
8160 RIGHTW 0,0,39,24
8220 REM ----- LAENGER WERDENDE WARTESCHLEIFE ----
8230 FOR J=1 TO 10*I
8240 NEXT
8250 NEXT
8260 REM ----- ANFLUGVORBEREITUNG ----------------
8270 B$=C$
8280 L=1
8300 RETURN
8500 REM ****************************************
8510 REM *   BEHANDLUNG MISSGLUECKTE LANDUNG     *
8520 REM ****************************************
8540 REM ------ BLINKEN SPRITE / BASIS -----------
8550 FLASH 8,10
8560 RLOCMOB 2,X0,Y0,0,1
8570 GOTO7200
9000 REM ****************************************
9010 REM *   BEHANDLUNG ERFOLGREICHE LANDUNG     *
9020 REM ****************************************
9030 L=0
9050 PRINT"SQQQQQQQQQQQ      πLANDUNG ERFOLGREICH
9060 FORI=1TO2000
9070 NEXT
9080 MOB SET 2,15,1,0,0
9100 PRINT"
9110 RETURN
9497 REM ****************************************
9498 REM *   MOB VERGROESSERN ODER VERKLEINERN   *
9499 REM ****************************************
9500 B$=C$
9510 GETA$
9520 IFA$=""THEN9510
9530 A=VAL(A$)
9540 IFA>3THEN9510
9550 GETNR$
9560 IFNR$=""THEN9550
9570 NR=VAL(NR$)
9580 IFNR>2THEN9550
9590 ON NR+1 GOTO 9600,9610,9620
9600 E0=A : RETURN
9610 E1=A : RETURN
9620 E2=A : RETURN
```

3.7 Variablen-Übersicht

RAKETE **100 - 9620**

Variablen:

Name	Typ	Bereich	Bedeutung
A$	G	1 Zeichen	Einlesen von Tastatur
A	H	0...7	Wert von A
B$	G	1 Zeichen	'Bewegungszeichen'
C$	G	1 Zeichen	Zwischenspeichern des Bewegungszeichens
E0	G	0...3	Vergrößerung des Abfangjägers
E1	G	0...3	Vergrößerung des Schusses
E2	G	0...3	Vergrößerung des Raumgleiters
G	H	1...9	Geschwindigkeitsangabe
G$	H	"1"..."9"	Geschwindigkeitsangabe
G0	G	2...18	Geschwindigkeit des Abfangjägers
G1	G	2...18	Geschwindigkeit des Schusses
G2	G	2...18	Geschwindigkeit des Raumgleiters
I,J	H	1...25 / 1...2000	Laufvariable
L	H	0,1	Merker für Landevorg.
NR	H	0...2	Spritenummer für Vergrößerung
NR$	H	"0"..."2"	Spritenummer für Vergrößerung
S	H	0,1	Merker für Schuss
T0	H	0,1	Merker für Kollision
X0	G	0...319	X-Koordinate Sprite 0
X1	G	0...319	X-Koordinate Sprite 1
X2	G	0...319	X-Koordinate Sprite 2
Y0	G	0...199	Y-Koordinate Sprite 0
Y1	G	0...199	Y-Koordinate Sprite 1
Y2	G	0...199	Y-Koordinate Sprite 2

Felder (Arrays):

Name	Dimen.	Typ	Bereich	Bedeutung

Unterprogrammaufrufe :

in	nach	Zweck
3006	6000	Sprite 2 bewegen
3006	7000	Kollision prüfen
3012	9030	geglückte Landung
3135	3500	Geschwindigkeit ändern
3136	8030	Landung vorbereiten
3138	9500	Sprites vergrößern / verkleinern

Verzweigungen nach außen :

in Ze	nach	Bedingung	Bemerkung
7276	RUN	Kollision	Neustart bei Koll.

4

Fehlermeldungen

4. Fehlermeldungen

Da das - uns vorliegende - Handbuch überhaupt nichts über Fehlermeldungen aussagt, wollen wir Ihnen - falls es Ihnen genauso geht - im folgenden die zwölf Simon's-spezifischen Fehlermeldungen näher erläutern. Wichtig: Die Fehlermeldungen von Simon's Basic werden nicht durch ON ERROR abgefangen. Die nachfolgende Aufstellung benutzt die interne Fehlernummer als Sortierkriterium. Die interne Fehlernummer kann normalerweise nicht abgefragt werden.

(1) PROC NOT FOUND

(Prozedur nicht gefunden)

Bei Aufruf der Befehle CALL und EXEC wird im Programm nach dem Namen des hinter dem Befehl angegebenen Unterprogramms gesucht. Wird dieser im Programm nicht gefunden, so wird die Fehlermeldung 'PROC NOT FOUND' ausgegeben.

(2) INSERT TOO LARGE

(Eingefügter Text zu lang)

Dieser Fehler tritt bei der Zeichenreihenoperation INSERT auf. Logischerweise darf beim Einfügen die Position, ab der eingefügt werden soll, nicht größer sein als die zweite Zeichenreihe lang ist.

(3) STRING TO LARGE

(Zeichenreihe zu lang)

Irgend etwas scheint hier schief gegangen zu sein; sicherlich sollte die Fehlermeldung 'STRING TOO LARGE' heißen. Er ist das Gegenstück in Simon's Basic für 'STRING TOO LONG'. Dieser Fehler kann bei den Befehlen INST, INSERT, DUP auftreten, wenn die sich ergebende Zeichenreihe mehr als 255 Zeichen enthält.

(4) NOT BINARY CHAR

(Kein binäres Zeichen)

Bei den Zahlbefehlen müssen nach dem '%'-Zeichen genau acht Zeichen, die aus '0' oder '1' bestehen, auftreten,

ansonsten wird Fehler (4) produziert.

(5) NOT HEX CHAR
(Kein hexadezimales Zeichen)

Ähnlich Fehler (4) müssen hier dem '$'-Zeichen genau vier hexadezimale Zeichen ('0123456789ABCDEF') folgen.

(6) END PROC WITHOUT EXEC
(Prozedurende ohne Prozeduraufruf)

Diese Fehlermeldung entspricht dem RETURN WITHOUT GOSUB im normalen Basic, wo das Programm auf das Ende eines Unterprogrammes gelaufen ist ohne daß vorher eins aufgerufen wurde. Der Rechner erkennt diesen Fehler, weil er feststellt, daß der Stack leer ist.

(7) END LOOP WITHOUT LOOP
(Schleifenende ohne Schleifenbeginn)

Ähnlich wie (6) ist die Fehlermeldung 'END LOOP WITHOUT LOOP' die andere Version für das Ende einer Schleife ohne das Anfangskommando (NEXT WITHOUT FOR).

(8) UNTIL WITHOUT REPEAT
(Schleifenende ohne Schleifenbeginn)

Analog (7) für den Schleifenbeginn 'REPEAT UNTIL'.

(9) STACK TOO LARGE
(Überlauf im Kellerspeicher)

In Kapitel 1 haben wir bereits erwähnt, daß Simon's Basic für die Befehle EXEC, REPEAT und LOOP jeweils einen eigenen Stack besitzt, der bis zu fünf Elemente enthalten kann. Wenn einer dieser drei Stacks überläuft, so erscheint Fehler (9).

(10) TOO FEW LINES
(zu wenig Zeilen - bei DESIGN)

Der Fehler (10) tritt nach dem DESIGN-Befehl auf, wenn entsprechend dem Parameter zu wenig Zeilen folgen. Bei DESIGN0 bzw. DESIGN1 müssen genau 21 beschreibende Zeilen folgen, bei DESIGN2 und DESIGN3 genau acht Zeilen.

(11) BAD CHAR FOR A MOB

(falsches Zeichen bei MOB-Definition)

Als beschreibende Zeichen nach dem DESIGN-Befehl sind ausschließlich zugelassen:

'Klammeraffe'	zur Zeilenanfangsmarkierung (und nur dort)
A,.,' '	Zeichen, die später keinen Punkt am Bildschirm erzeugen
B	Punkt in der ersten Farbe
C,D	Punkt in der 2. und 3. Farbe (jedoch nur nach DESIGN 1 oder DESIGN 3)

Alle anderen Zeichen führen zu dieser Fehlermeldung.

(12) BAD MODE

(Falscher Modus / allgemeiner Fehler)

Den Fehler 'BAD MODE' kann man als 'SYNTAX ERROR' von Simon's Basic bezeichnen. Er wird immer dann angesprochen, wenn Unstimmigkeiten vorliegen und keiner der Fehler (1) bis (11) vorliegt. Manchmal wird anstatt 'BAD MODE' auch 'DS-CBM' ausgegeben. Die Bedeutung ist ähnlich.

N.B.: Das Statusbyte ST wird auf acht gesetzt, wenn bei einem Grafikbefehl unzulässige Punkt-Koordinaten angegeben werden. D.h. Sie brauchen die umständliche Abfrage in Basic bezüglich Korrektheit der Koordinaten zunächst nur auf das Statusbyte zu beschränken, und wenn dieses acht ist eine genaue Untersuchung der Koordinaten vornehmen.

5

Musik an Beispielen

5. Musik am Beispiel

Im Handbuch von Simon's Basic ist zwar ein Musikbeispiel abgebildet, aber anhand dieser Vorgabe könnte man nur einstimmig spielen. In dem nachfolgenden Programm wollen wir Ihnen zeigen, wie mit Simon's Basic auch alle drei Stimmen genutzt werden können. Um die Eingabe zu vereinfachen, kann jede Stimme - wie bei dem Beispiel im Handbuch auch - getrennt eingegeben werden. Diese werden anschließend mit einer eigenen Prozedur gemischt. Eine weitere Prozedur beschäftigt sich mit der Klangformung und eine dritte mit dem tatsächlichen Spiel, da der eigentliche Musikbefehl nur eine Zeichenreihe von maximal 255 Zeichen zuläßt.

Im weiteren wurden in diesem Kapitel für die Unterprogramme die Simon's-Befehle EXEC, PROC, REPEAT...UNTIL und CALL verwendet.

Da die Notendauerumschaltung jeweils für alle drei Stimmen gilt und das dauernde Umschalten der Noten einerseits Speicherplatz (in den Zeichenreihen) und mühsame Tipparbeit kostet, andererseits zwei aufeinanderfolgende gleiche Noten akustisch nicht zu unterscheiden sind, geben wir für das gesamte Stück eine einzige Notendauer vor. In unserem Beispiel sind es Viertel, d.h. halbe Noten erfordern die doppelte und ganze Noten die vierfache Eingabe der Tonhöhe. Sie werden sehr schnell feststellen, daß dies die Eingabe wesentlich erleichtert.

Zunächst also wird in Zeile 100 die Notendauer auf eine Viertelnote festgelegt. Die Zeile 110 bis 130 beinhalten den ersten Teil der drei Stimmen, da in den Zeilen nicht mehr Noten erfaßt werden können. Diese drei Stimmen werden durch den Prozeduraufruf in Zeile 140 zunächst ineinandergemischt und anschließend erfolgt die Eingabe des zweiten Teils für alle drei Stimmen, die in Zeile 240 wieder zusammengemischt werden. Die beiden letzten Prozeduraufrufe sind dann für den Klang und das eigentliche Spiel zuständig.

Mischen

Die Prozedur Mischen besteht selbst wieder aus zwei Prozeduren: MISCH und M1. In MISCH wird je eine Note für die einzelnen Stimmen abgetrennt und in dem Musikstring MU$(M)

```
100 D1$="▌█"
110 A1$="B4E5‾5E5_5G5_5G5A5B5B5B5B5B5B5D6D6B5B5B5B5B5E6E6
                                                D6B5B5B5B5B5"
120 A2$="PE4G4E4G4E4G4E4G4G3G4G3G4G3G4G3G4G3G4G3G4G3G4G3G4
                                          G3G4G3G4G3G4G3G4E4G4"
130 A3$="PPB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4PB4
                                                 PB4PB4PB4PB4P"
140 EXEC MISCH
210 A1$=A1$+"A5G5E5E5E5E5E5E5A5A5G5E5E5E5E5E5E5]G"
220 A2$=A2$+"E4G4E4G4E4G4E4G4E4G4E4G4E4]G"
230 A3$=A3$+"B4PB4PB4PB4P]G"
240 EXEC MISCH
400 EXEC KLANG
410 EXEC SPIEL
999 END
1000 PROC MISCH
1010 REPEAT
1020 IFA1$>""THENMU$(M)=MU$(M)+"]1"+D1$:A0$=A1$:GOSUB1500
                                                      :A1$=A0$
1030 IFA2$>""THENMU$(M)=MU$(M)+"]2":A0$=A2$:GOSUB1500
                                                      :A2$=A0$
1040 IFA3$>""THENMU$(M)=MU$(M)+"]3":A0$=A3$:GOSUB1500
                                                      :A3$=A0$
1050 IFLEN(MU$(M))>240THENM=M+1
1060 UNTIL A1$="" OR A2$="" OR A3$=""
1070 END PROC
1500 PROC M1
1510 AA$=LEFT$(A0$,1)
1520 A0$=MID$(A0$,2)
1530 IFPLACE(AA$,"[illegible]")THEND0$=AA$: CALL M1
1540 IFPLACE(AA$,"CDEFGAB‾‾_|♠"┌┴▖| ┌▘")THENMU$(M)=
                                         MU$(M)+AA$: CALL M1
1550 IFPLACE(AA$,"01234567 PZ")THENMU$(M)=MU$(M)+AA$
1560 RETURN
2000 PROC SPIEL
2010 FORI=0TOM
2020 MUSIC5,MU$(I)
2030 PLAY 1
2040 NEXT
2050 VOL0
2060 END PROC
3000 PROC KLANG
3010 ENVELOPE 1,3,3,10,5
3020 ENVELOPE 2,4,4,10,5
3030 ENVELOPE 3,4,4,10,5
3040 WAVE1,01000000
3050 WAVE2,00010000
3060 WAVE3,00010000
3070 POKE SOUND+3,15
3100 VOL15
3110 END PROC
```

abgelegt. Dazu wird das Unterprogramm ab Zeile 1500 für das eigentliche Abtrennen aufgerufen. Hier wird deutlich, daß auch bei Verwendung von Simon's-Prozeduren weiterhin Unterprogramme mit GOSUB aufgerufen werden können. Da bei dreistimmigem Spiel sicherlich mehr als 255 Zeichen benötigt werden, um ein annehmbares Musikstück zu spielen, wurde die Zeichenreihe für das eigentliche Spiel als Feld (Array) ausgelegt. Durch das Abtrennen ab Zeile 1500 können pro Stimme nie mehr als fünf Zeichen an die aktuelle Zeichenreihe zum Spielen angehängt werden. Deshalb befindet sich in Zeile 1050 die Abfrage, ob die Zeichenreihe bereits die Länge 240 hat. In diesem Falle wird aus dem Array das nächste Element zur weiteren Verarbeitung herangezogen. Das Endekriterium für das Mischen ist erfüllt, wenn eine der drei Einzelstimmen (bzw. ihre Zeichenreihe) leer ist.

In der Prozedur ab Zeile 1500 geschieht das eigentliche abtrennen. Dabei wird zunächst das erste Zeichen an AA$ übergeben und der Rest in AO$ gespeichert. Wer seine Notendauer während des Stückes doch ändern will kann in Zeile 1530 die aktuelle Notendauer erfragen, die dann an die Variable DO$ übergeben wird. Ist das erste Zeichen ein Buchstabe oder ein anderes zulässiges Notenzeichen, so wird dieses Zeichen an den aktuellen Musikstring angehangen und das Unterprogramm erneut aufgerufen. Liegt eine Zahl oder ein Pausenzeichen (P oder Z) vor, so wird nur noch dies an den aktuellen Musikstring angehangen und das Unterprogramm verlassen.

Spielen

Da der Musikbefehl immer nur eine Zeichenreihe spielen kann, beinhaltet diese Prozedur lediglich eine Schleife, in der alle zusammengesetzten Musikstrings nacheinander gespielt werden. Dazu ist der Befehl PLAY1 unbedingt erforderlich. Zum Abschluß wird noch die Lautstärke ausgeschaltet.

Klang

In dem Unterprogramm zur Bestimmung des Klanges werden zunächst die drei Hüllkurven entsprechend mit den Werten für Attack, Decay, Sustain und Release besetzt (der erste Parameter ist die Stimme). Dann werden noch die einzelnen Wellenformen eingeschaltet, wobei wir für die erste Stimme die Rechteckform gewählt haben und für die Begleitstimmen die Dreieckschwingung. Da bei einer Rechteckschwingung unß-dingt das Pulsverhältnis angegeben werden muß, ist der

Befehl in Zeile 3070 noch erforderlich, um die erste Stimme hörbar zu machen. Weil das Pulsverhältnis in zwei Byte abgespeichert wird, werden mit Zeile 3070 nur die letzten vier Bit des Higher Byte gesetzt. Das Pulsverhältnis beträgt also 15*256, fast die gesamte Pulsbreite. Wir erinnern hier nochmal an Band 3, wo wir festgestellt haben, daß die Pulsbreite sich in ein Lower Byte und Higher Byte gliedert, wobei im Higher Byte jedoch nur die letzten vier Bit signifikant sind (das Pulsverhältnis ist eine 12-Bit Zahl).

Als letztes wird noch die Lautstärke auf maximale Höhe gesetzt. Beachten Sie, daß die Prozedur Klang immer vor der Prozedur Spiel aufgerufen werden muß, da Sie sonst um Ihr akustisches Vergnügen gebracht werden. Ach ja, haben Sie das Lied erkannt? El Condor Pasa!

6

Kommentiertes Assembler-Listing

6. Kommentiertes Assembler-Listing

Um denjenigen, die Simon's Basic noch verbessern wollen oder auch nur einige Tips für die eigene Programmierung ableiten wollen ein geeignetes Hilfsmittel zu geben stellen wir in diesem Kapitel das kommentierte Assembler-Listing vor.

Auch Simon's Basic ist Software, Software ist Handarbeit und Handarbeit ist nie fehlerfrei. Der Programmierer oder das Team, das Software solchen Umfanges fehlerfrei schreibt muß wohl erst noch geboren werden, oder selbst ein Computer sein. Wer Fehler beseitigen, weitere Befehle einbauen oder einige Befehle austauschen oder löschen will (um Speicherplatz frei zu bekommen) dem dürfte besonders Kapitel 6.2 willkommen sein.

Zunächst soll jedoch noch einiges zum Kommentar selbst gesagt werden, und anschließend bringen wir noch eine Übersicht der verwendeten Symbole (nach Namen und Adressen geordnet. Den Abschluß von Kapitel 6 bildet eine Übersicht der von Simon's Basic verwendeten Adressen der Zero-Page.

Übrigens: Grundlage für diese Arbeit war der Dissassembler aus Band 4, der noch leicht modifiziert wurde.

6.1 Allgemeines / Bemerkungen zum Kommentar

'Bemerkungen zum Kommentar' hört sich fast an, als hätte man es mit Gesetzen zu tun. Dies ist zwar nicht der Fall, jedoch scheint es aufgrund des langen Listing, geraten, etwas zur prinzipiellen Vorgehensweise zu sagen.

6.1.1 Symbolbenennung

Zur Darstellung von Unterprogramm-Labels und Hilfszellen wurden statt der üblichen hexadezimalen Adressen **Symbole** verwendet, dadurch wird die Lesbarkeit des Listings wesentlich erhöht. Manchmal interessieren aber auch die tatsächlichen Werte der Labels (Marken). Diese kann man in der Tabelle in Kapitel 6.3 nachschlagen. Schneller geht es natürlich, wenn man sich die Code-Spalten (die Spalten hinter der Adresse) genauer ansieht. Den **Wert des Symbols** erhält man einfach durch Vertauschen der zweiten und der dritten Spalte. Ein Beispiel:

```
800D  20 2C 81   JSR PSPEIBE   ;Basic-Rom ein
```

Die Adresse von PSPEIBE ist also $812C.

Symbole, die Namen der Art '**SYMBOL**','**SYMBOL+1**' tragen, bezeichnen Lower- und Higher-Byte eines 16-Bit-Wertes. Bei den Bemerkungen ist dann oft nur der Name ohne '+1' erklärt.

Symbolnamen, die '**TAB**' enthalten, bezeichnen eine Tabelle (d.h. wenigstens 3 gleichartige Werte).

Symbolnamen, die mit '**S**' anfangen, aber auch ohne das 'S' Sinn geben (z.B. SCHKCOM), bezeichnen einen für Simon's Basic veränderten Aufruf der Routine ohne das 'S' (hier also CHKCOM) oder die Simon's Version einer auch im normalen Basic vorhandenen ähnlichen Routine oder einer Vorgänger-Routine davon (z.B. SNMI; SIRQ).

Symbolnamen, die mit '1' oder '2' aufhören, sind Fortsetzungen oder Abwandlungen.

Den **Zero-Page-Adressen** wurden keine Symbole zugeordnet, da deren Bedeutung meist nicht allgemein festgelegt werden kann. Eine Tabelle der wichtigsten Bedeutungen für Simon's Basic von Zellen in der Zero-Page ist in Kapitel 6.4 wiedergegeben.

Symbole, die mit '**BEF**' beginnen, bezeichnen die Startadresse der Befehlssequenz des entsprechenden Simon's-Basic-Befehls. Die Befehle INV, MOVE, UPW, DOWNW, LEFTW, RIGHTW, UPB, DOWNB, LEFTB, RIGHTB haben eine gemeinsame Adresse 'MOVEBEF'; erst später wird zu den einzelnen Routinen verzweigt. Deshalb fehlen in der Symboltabelle Bezeichnungen wie 'BEFINV'.

Symbole, die mit '**FN**' beginnen bezeichnen eine Simon's-Funktion.

6.1.2 Zur Kommentierung

Meistens ist der Kommentar weggelassen worden, wenn die Bedeutung der Befehle sofort einsichtig ist oder sich aus den Bemerkungen in den darüber liegenden Zeilen leicht ergibt. Insbesondere wird eine analoge Bedeutung von Lower- und Higher-Byte eines Ausdrucks nur einmal erklärt. Oft erscheinen im Kommentar auch die Befehlsfolgen für Lower- und Higher-Byte zusammengefasst.

Nach **JMP**-Befehlen, nach **RTS** und nach einem logischen Absatz sowie vor **wichtigen Sprungzielen** ist eine Leerzeile eingefügt.

Bei **relativen Sprüngen** ist als Kommentar oft nur die Bedingung angegeben, unter der verzweigt wird. Zum Beispiel:

```
BEQ  $AAAA     ;Zeilenende ?
```

Es wird nach $AAAA verzweigt, wenn ein Zeilenende erreicht wurde.

Sprünge, die wiederholt ausgeführt werden, etwa bis ein Zähler abgelaufen ist, sind oft mit '**Schleife**' kommentiert. Steht dieser Kommentar hinter einem JMP-Befehl, so wird die Schleife an einer anderen Stelle, die weiter oben angegeben ist, verlassen.

Die Wörter '**MOB**' und '**Sprite**' sind Synonyme.

'**akt.**' ist die Abkürzung für 'aktuell' und wird meist verwendet, wenn ein veränderlicher Zeiger (z.B. der Basic-Programmzeiger) auf ein Byte zeigt. Dieses Byte wird dann mit 'akt. Byte' bezeichnet.

Die **Dereferenzierungen** wurden relativ leger gehandhabt. Z.B. bedeutet der Kommentar '$A4 = $61 + COPZAEY', daß sich die Speicherzelle $A4 mit der Summe aus dem Inhalt der Speicherzelle $61 und der Zelle COPZAEY ergibt. Analog wurde kommentiert, wenn ein Zellenpaar angesprochen wird: der Kommentar '$09,$0A = $5F,$60 + PZEIG' weist auf eine 16-Bit-Addition hin, deren erster Summand aus den Zellen $5F und $60 geholt wird. Das Lower-Byte des zweiten Summanden wird aus PZEIG geholt, das Higher-Byte davon ist 0.

6.1.3 Speicherverteilung (RAM-Belegung)

$0000-$3FFF	Belegung wie bei normalem Basic
$0400-$7FFF	Video-RAM bei normalem Betrieb
$0800-$7999	Speicher für Anwenderprogramm und Variablen
$8000-$BFFF	Simon's Basic
$C000-$C3FF	Video-RAM bei Hires (1. und 2. Farbe)
$C400-$C4FF	frei
$C500-$CBFF	Hilfszellen für Simon's Basic
$CC00-$CFFF	Video-RAM nach MEM
$D000-$DFFF	I/O und Farb-RAM
$E000-$FF3F	Grafik-Speicher bzw. Zeichengen. nach MEM
$FF40-$FFF9	(freies RAM)

```
$FFFA,$FFFB   NMI-Vektor (im RAM auf $8118 gerichtet)
$FFFC-$FFFF   im RAM nicht belegt
```

Da im Bereich von $A000 bis $BFFF das Basic-ROM liegt, muß dies ausgeschaltet werden, wenn ein Simon's-Basic-Befehl ausgeführt wird. Die Simon's Funktionen liegen allesamt im Bereich bis $9FFF, sodaß hier nicht umgeschaltet werden braucht.

Der Grafik-Speicher liegt unter dem Betriebssystem-ROM, so daß dieses ausgeschaltet werden muß, wenn das Grafik-RAM bearbeitet wird, z.B. bei der TEST-Funktion.

6.1.4 Aufbau der Simon's- Befehle und Funktionen

Alle Simon's Befehle und Funktionen sind im Programmspeicher durch das Byte $64 gekennzeichnet, dem die Nummer des Befehls (1 Byte) folgt.

Für die Simon's-Befehle existiert eine Sprungtabelle, für die Funktionen nicht. In der Sprungtabelle für die Befehle steht an den Stellen, die den Funktionen entsprechen, die Adresse von 'BADMODE', wodurch eine Fehlermeldung ausgegeben wird, wenn versucht wird, eine Funktion als Befehl aufzurufen.

Typischer Aufbau eines Befehls von Simon's Basic:

i) Code überlesen (Codenr. des Befehls); dies geschieht z.B. mit JSR INCBASBZ

ii) Parameter holen; z.B. mit JSR SGETADR (holt 16-bit-Wert)

iii) eigentliche Befehlsbearbeitung

iv) Befehl beenden; z.B. mit JMP ENDSMB

Werden keine Parameter gebraucht, so können i) und iv) zusammengefasst werden. Punkt i) entfällt, und an Stelle von 'JMP ENDSMB' tritt 'JMP BEFO'.

Ist der erste Parameter ein Byte-Wert, können i) und ii) zusammengefasst werden zu 'JMP SGETBYTN'. SGETBYTN überliest ein Zeichen, bevor es den Byte-Parameter auswertet.

Typischer Aufbau einer Funktion von Simon's Basic:

Da keine Sprungtabelle existiert, steht zu Beginn die Abfrage, ob diese Funktion auch wirklich aufgerufen werden soll, d.h. es werden alle Funktionen nacheinander bis zur gefundenen durchsucht. Dann muß - wie bei den Befehlen - der Befehlscode überlesen werden.

Die Parameter stehen bei Funktionen in Klammern. Deshalb steht vor dem Parameterholen ein Aufruf von CHKKLA (folgt Klammer auf ?) und danach ein Aufruf von CHKKLZ (folgt Klammer zu ?).

Es gibt String-Funktionen und numerische Funktionen. Der Typ der Funktion wird dem normalen Basic durch das Flag in Zelle $0D mitgeteilt. ($00 = numerisch, $FF = String). Am Ende der Funktions-Routine wird dieses Flag gesetzt.

Das Ergebnis der Funktion muß bei numerischen Funktionen im Fließkomma-Akkumulator (kurz: FAC) übergeben werden; bei String-Funktionen muß der Deskriptor (Länge und Adresse) des Strings auf den Stringstapel gelegt werden. Dies wird durch die Basic-Routine PUSHSTR erledigt.

6.1.5 Interrupt-Steuerungen

Per Interrupt werden diverse Funktionen gesteuert:

- PAUSE

  Der Sekundenzähler für den Pause-Befehl wird durch Interrupt hochgezählt.

- BFLASH

- FLASH

- KEY

- PLAY

  Der MUSIC-String wird durch eine Interrupt-Routine abgearbeitet.

Der Interrupt-Vektor wird dazu auf $9694 gestellt. Die jeweiligen Funktionen werden ausgeführt, wenn die entsprechenden Flags gesetzt sind.

Für den NMI steht eine besondere Adresse im RAM zur Vefügung, wenn gerade das KERNAL ausgeschaltet ist. Bei dieser Adresse wird das KERNAL wieder eingeschaltet und zur normalen NMI-Routine verzweigt.

6.2 Listing

```
STARTVEC:
 8000  0A 80              $800A    Startvektor für Reset
SNMIVEC:
 8002  FF 82              $82FF    Vektor für NMI

MODULTEXT:
 8004  C3 C2 CD 38 30     "CBM80"  ;für Modulerkennung

 8009  FF           BYT   $FF      ;(unsinnig)

SSTART:
 800A  4C 47 81     JMP   BEFCOLD  ;zum COLD-Befehl

*** Es folgen einige Aufrufe von Basic-Routinen, wobei
*** zunächst mit PSPEIBE  der momentane Wert des Prozes-
*** sorports gespeichert und das Basic-ROM eingeschaltet
*** wird. Dann wird die entsprechende Routine aufgerufen
*** und anschließend der alte Wert im Prozessorport
*** wieder hergestellt (mittels PRESTOR).

SFRMNUM:
 800D  20 2C 81     JSR   PSPEIBE  ;Basic-Rom ein
 8010  20 8A AD     JSR   $AD8A    ;holt numerischen Ausdruck
 8013  4C 3B 81     JMP   PRESTOR  ;stellt ROM-Schalter zur.

SFACADR:
 8016  20 2C 81     JSR   PSPEIBE
 8019  20 F7 B7     JSR   $B7F7    ;wandelt FAC in Adreßform.
 801C  4C 3B 81     JMP   PRESTOR

SSIN:
 801F  20 2C 81     JSR   PSPEIBE
 8022  20 6B E2     JSR   $E26B    ;Basic-Funktion SIN
 8025  4C 3B 81     JMP   PRESTOR

SCOS:
 8028  20 2C 81     JSR   PSPEIBE
 802B  20 64 E2     JSR   $E264    ;Basic-Funktion COS
 802E  4C 3B 81     JMP   PRESTOR

SCHKCOM:
 8031  20 2C 81     JSR   PSPEIBE
 8034  20 FD AE     JSR   CHKCOM   ;prüft auf Komma
 8037  4C 3B 81     JMP   PRESTOR

SFRMEVL:
 803A  20 2C 81     JSR   PSPEIBE
 803D  20 9E AD     JSR   $AD9E    ;holt beliebigen Ausdruck
 8040  4C 3B 81     JMP   PRESTOR
```

```
SFRESTR:
 8043   20 2C 81    JSR   PSPEIBE
 8046   20 A3 B6    JSR   $B6A3      ;holt Stringparameter
 8049   4C 3B 81    JMP   PRESTOR

SINTOUT:
 804C   20 2C 81    JSR   PSPEIBE
 804F   20 D1 BD    JSR   $BDD1      ;gibt Ganzzahl aus
 8052   4C 3B 81    JMP   PRESTOR

SGOTO1:
 8055   20 2C 81    JSR   PSPEIBE
 8058   20 A3 A8    JSR   $A8A3      ;springt zu Zeile($14,$15)
 805B   4C 3B 81    JMP   PRESTOR

SCHKZEI:
 805E   20 2C 81    JSR   PSPEIBE
 8061   20 FF AE    JSR   $AEFF      ;prüft auf Zeichen im Akku
 8064   4C 3B 81    JMP   PRESTOR

SCHKCOM1:
 8067   20 2C 81    JSR   PSPEIBE
 806A   20 FD AE    JSR   CHKCOM     ;prüft auf Komma
 806D   4C 3B 81    JMP   PRESTOR

SSTRPZ:
 8070   20 2C 81    JSR   PSPEIBE
 8073   20 13 A6    JSR   $A613      ;berechnet Zeilenstartadr.
 8076   4C 3B 81    JMP   PRESTOR

SASCFLP:
 8079   4C F3 BC    JMP   $BCF3      ;wandelt Text nach Fließk.

SGETARI1:
 807C   4C 92 AE    JMP   $AE92      ;holt arithmet. Element

SCHKKLZ:
 807F   20 2C 81    JSR   PSPEIBE
 8082   20 F7 AE    JSR   CHKKLZ     ;prüft auf 'Klammer zu'
 8085   4C 3B 81    JMP   PRESTOR

SXFLP:
 8088   20 2C 81    JSR   PSPEIBE
 808B   20 49 BC    JSR   XFLP       ;wand. nach FAC (Exp in X)
 808E   4C 3B 81    JMP   PRESTOR

SFACARG:
 8091   20 2C 81    JSR   PSPEIBE
 8094   20 0C BC    JSR   FACARG     ;FAC nach ARG übertr.
 8097   4C 3B 81    JMP   PRESTOR
```

```
SFMULT:
 809A  20 2C 81   JSR  PSPEIBE
 809D  20 30 BA   JSR  $BA30    ;Fließk.-Multiplikation
 80A0  4C 3B 81   JMP  PRESTOR

SFLDFACK:
 80A3  20 2C 81   JSR  PSPEIBE
 80A6  20 A2 BB   JSR  $BBA2    ;lädt FAC mit Konstante
 80A9  4C 3B 81   JMP  PRESTOR

SFDIV:
 80AC  20 2C 81   JSR  PSPEIBE
 80AF  20 14 BB   JSR  $BB14    ;Fließk.-Division
 80B2  4C 3B 81   JMP  PRESTOR

SFACASC:
 80B5  20 2C 81   JSR  PSPEIBE
 80B8  20 DF BD   JSR  $BDDF    ;wandelt FAC in Text um
 80BB  4C 3B 81   JMP  PRESTOR

SGETVAR:
 80BE  20 2C 81   JSR  PSPEIBE
 80C1  20 28 AF   JSR  $AF28    ;holt Wert von Variable
 80C4  4C 3B 81   JMP  PRESTOR

STESTDIRM:
 80C7  20 2C 81   JSR  PSPEIBE
 80CA  20 A6 B3   JSR  $B3A6    ;prüft auf Direktmodus
 80CD  4C 3B 81   JMP  PRESTOR

SCLRCH:
 80D0  20 2C 81   JSR  PSPEIBE
 80D3  20 B5 AB   JSR  $ABB5    ;schließt alle Kanäle
 80D6  4C 3B 81   JMP  PRESTOR

SREM:
 80D9  20 2C 81   JSR  PSPEIBE
 80DC  20 F8 A8   JSR  $A8F8    ;entspricht Basic-Bef. REM
 80DF  4C 3B 81   JMP  PRESTOR

SNEXTTR:
 80E2  20 2C 81   JSR  PSPEIBE
 80E5  20 06 A9   JSR  $A906    ;nächstes Trennz. suchen
 80E8  4C 3B 81   JMP  PRESTOR

SINPUT1:
 80EB  20 2C 81   JSR  PSPEIBE
 80EE  20 0D AC   JSR  $AC0D    ;entspr. Basic-Bef. INPUT
 80F1  4C 3B 81   JMP  PRESTOR
```

```
SPUFOCR:
 80F4   20 2C 81    JSR   PSPEIBE
 80F7   20 CA AA    JSR   $AACA      ;Puffer mit 0 abschließen
                                     ;CR ausgeben
 80FA   4C 3B 81    JMP   PRESTOR

SBZPLY:
 80FD   20 2C 81    JSR   PSPEIBE
 8100   20 FB A8    JSR   $A8FB      ;Programmzähler plus Y
 8103   4C 3B 81    JMP   PRESTOR

SGETBYT1:
 8106   20 2C 81    JSR   PSPEIBE
 8109   20 9E B7    JSR   $B79E      ;holt Byte-Wert
 810C   4C 3B 81    JMP   PRESTOR

SGOTO11:
 810F   20 2C 81    JSR   PSPEIBE
 8112   20 A3 A8    JSR   $A8A3      ;springt zu Zeile($14,$15)
 8115   4C 3B 81    JMP   PRESTOR

SSMNI:
 8118   A5 01       LDA   $01
 811A   8D 8B C5    STA   PSPEINMI    ;Merker setzen
 811D   20 0E 94    JSR   KERROMEIN   ;KERNAL-Rom einschalten
 8120   4C 43 FE    JMP   $FE43       ;zum Standard-NMI

SOPEN:
 8123   20 2C 81    JSR   PSPEIBE
 8126   20 BE E1    JSR   $E1BE       ;Basic-Befehl OPEN
 8129   4C 3B 81    JMP   PRESTOR


PSPEIBE :
 812C   8D 7D C5    STA   ZWSPEI1    ;sichert momentanen Akku
 812F   A5 01       LDA   $01        ;holt Prozessorport
 8131   8D 96 C5    STA   PORTSPEI   ;sichert diesen Wert
 8134   20 F3 82    JSR   BASROMEI   ;schaltet Basic-Rom ein
 8137   AD 7D C5    LDA   ZWSPEI1    ;holt Akku wieder
 813A   60          RTS

PRESTOR:
 813B   8D 7D C5    STA   ZWSPEI1    ;sichert momentanen Akku
 813E   AD 96 C5    LDA   PORTSPEI   ;holt alten Wert für Port
 8141   85 01       STA   $01        ;speichert Port
 8143   AD 7D C5    LDA   ZWSPEI1    ;holt Akku wieder
 8146   60          RTS
```

```
BEFCOLD:
 8147   20 F3 82    JSR   BASROMEI ;Basic-Rom ein
 814A   20 53 E4    JSR   $E453    ;Basic-Vektoren laden
 814D   20 BF E3    JSR   $E3BF    ;Ram f. Basic vorbereiten
 8150   A9 0F       LDA   #$0F     ;Hellgrau
 8152   8D 21 D0    STA   VICHIFAR ;als Hintergrundfarbe
 8155   A9 06       LDA   #$06     ;Blau
 8157   8D 20 D0    STA   VICRAFAR ;als Rahmenfarbe

 815A   A9 00       LDA   #$00     ;Null in
 815C   AA          TAX            ;
 815D   9D 00 C3    STA   $C300,X  ;Hilfszellenbereich
 8160   9D 00 C4    STA   $C400,X  ;($C300-CBFF)
 8163   9D 00 C5    STA   $C500,X  ;schreiben
 8166   9D 00 C6    STA   $C600,X
 8169   9D 00 C7    STA   $C700,X
 816C   9D 00 C8    STA   $C800,X
 816F   9D 00 C9    STA   $C900,X
 8172   9D 00 CA    STA   $CA00,X
 8175   9D 00 CB    STA   $CB00,X
 8178   E8          INX            ;jeweils nächstes Byte
 8179   D0 E2       BNE   $815D    ;Schleife

 817B   A9 7A       LDA   #$7A     ;Vektor ($0308,$0309)
 817D   8D 08 03    STA   $0308    ;'Basic-Befehl ausf.' auf
 8180   A9 91       LDA   #$91     ;$917A = SBASBEF   setzen
 8182   8D 09 03    STA   $0309
 8185   A9 8D       LDA   #$8D     ;Vektor ($030A,$030B)
 8187   8D 0A 03    STA   $030A    ;'arithm. Elem. holen' auf
 818A   A9 8C       LDA   #$8C     ;$8C8D = SGETARI    setzen
 818C   8D 0B 03    STA   $030B
 818F   BD 08 82    LDA   MELDUNG,X  ;Einschaltmeldung
 8192   F0 06       BEQ   $819A    ;Endekriterium: Akku=0
 8194   20 D2 FF    JSR   BSOUT    ;Zeichen ausgeben
 8197   E8          INX            ;nächstes Zeichen
 8198   D0 F5       BNE   $818F    ;unbed. Sprung z. Schleife
 819A   A9 80       LDA   #$80
 819C   85 38       STA   $38      ;Basic-RAM-Ende  ($37,$38)
 819E   85 34       STA   $34      ;String-Untergr. ($33,$34)
 81A0   A9 00       LDA   #$00     ;auf $8000 setzen
 81A2   85 33       STA   $33
 81A4   85 37       STA   $37
 81A6   A9 82       LDA   #$82     ;Vektor ($0306,$0307)
 81A8   8D 07 03    STA   $0307    ;'in Klartext umwand.' auf
 81AB   A9 40       LDA   #$40     ;$8240 = SLIST      setzen
 81AD   8D 06 03    STA   $0306
 81B0   A9 82       LDA   #$82     ;Vektor ($0304,$0305)
 81B2   8D 05 03    STA   $0305    ;'Zeile umwandeln'    auf
 81B5   A9 34       LDA   #$34     ;$8234 = ZEIUMW     setzen
 81B7   8D 04 03    STA   $0304
```

```
 81BA   A9 9E       LDA   #$9E       ;Vektor ($0300,$0301)
 81BC   8D 01 03    STA   $0301      ;'Basic-Warmstart'    auf
 81BF   A9 9C       LDA   #$9C       ;$9E9C = SWARM      setzen
 81C1   8D 00 03    STA   $0300
 81C4   A9 82       LDA   #$82       ;Vektor ($0302,$0303)
 81C6   8D 03 03    STA   $0303      ;'Interpreterschleife' auf
 81C9   A9 5C       LDA   #$5C       ;$825C = SLOOP      setzen
 81CB   8D 02 03    STA   $0302
 81CE   A9 82       LDA   #$82       ;Vektor ($0318,$0319)
 81D0   8D 19 03    STA   $0319      ;'NMI-Vektor'      auf
 81D3   A9 FA       LDA   #$FA       ;$82FA = SNMI   setzen
 81D5   8D 18 03    STA   $0318
 81D8   A9 83       LDA   #$83       ;Vektor ($0316,$0317)
 81DA   8D 17 03    STA   $0317      ;'BRK-Vektor'      auf
 81DD   A9 14       LDA   #$14       ;$8314 = SBRK   setzen
 81DF   8D 16 03    STA   $0316
 81E2   A5 2B       LDA   $2B
 81E4   A4 2C       LDY   $2C
 81E6   20 08 A4    JSR   $A408      ;Platz bis (A/Y) schaffen
 81E9   20 30 E4    JSR   $E430      ;'xx bytes free' ausgeben
 81EC   A9 18       LDA   #$18       ;Vektor $FFFA,$FFFB in RAM
 81EE   8D FA FF    STA   $FFFA      ;'Hardware-NMI-Vektor' auf
 81F1   A9 81       LDA   #$81       ;$8118 = SSNMI      setzen
 81F3   8D FB FF    STA   $FFFB
 81F6   A2 FB       LDX   #$FB       ;$FB
 81F8   9A          TXS              ;in Stackpointer schreiben
 81F9   4C 86 E3    JMP   WARM       ;zum Basic-Warmstart $E386

SGETBYTC:
 81FC   20 31 80    JSR   SCHKCOM    ;prüft auf Komma
 81FF   4C 06 81    JMP   SGETBYT1   ;holt Byte-Wert

SGETBYTN:
 8202   20 DB 83    JSR   INCBASBZ   ;überliest 1 Zeichen
 8205   4C 06 81    JMP   SGETBYT1   ;holt Byte-Wert


MELDUNG:
 8208   90 93 2A 2A 2A 20 45 58 50     (black) (clr) *** exp
 8211   41 4E 44 45 44 20 43 42 4D     anded cbm
 821a   20 56 32 20 42 41 53 49 43      v2 basic
 8223   20 2A 2A 2A 0D 0D 00             *** (cr) (cr)


ENDSMB:                              ;dient als Abschluß eines
                                     ;Simon's Befehls
 822A   20 0E 94    JSR   KERROMEI   ;KERNAL-ROM einschalten
 822B   58          CLI              ;Interrupts ermöglichen
 822E   20 F3 82    JSR   BASROMEI   ;Basic-ROM ein
 8231   4C AE A7    JMP   $A7AE      ;zur (normalen) Basic-
                                     ;Interpreterschleife
```

```
ZEIUMW:                              ;wandelt eine Zeile in
                                     ;Interpretercode
 8234  20 EC 82   JSR  BASROMAU ;Basic-ROM aus
 8237  20 79 AB   JSR  ZEIUMW1  ;Umw. f. Simon's Befehle
 823A  20 F3 82   JSR  BASROMEI ;Basic-ROM ein
 823D  4C 7C A5   JMP  $A57C    ;zur (normalen) Umwandl.

SLIST:
 8240  08         PHP
 8241  48         PHA
 8242  20 EC 82   JSR  BASROMAU ;Basic aus
 8245  68         PLA
 8246  28         PLP
 8247  4C CB BE   JMP  SLIST3   ;Simon's Befehl dekodieren

SLIST1:
 824A  48         PHA
 824B  20 F3 82   JSR  BASROMEI ;Basic ein
 824E  68         PLA
 824F  28         PLP
 8250  4C 1A A7   JMP  $A71A    ;zur Basic-LIST-Routine

SLIST2:
 8253  48         PHA
 8254  20 F3 82   JSR  BASROMEI ;Basic ein
 8257  68         PLA
 8258  28         PLP
 8259  4C F6 A6   JMP  $A6F6    ;Basic-LIST-Routine:
                                ;Zeichen bis Zeilenende
                                ;dekodieren und ausgeben

SLOOP:                          ;Eingabe-Schleife
 825C  78         SEI
 825D  A9 96      LDA  #$96     ;IRQ-Vektor ($0314,$0315)
 825F  8D 15 03   STA  $0315    ;auf $9694 = SIRQ
 8262  A9 94      LDA  #$94     ;stellen
 8264  8D 14 03   STA  $0314
 8267  8D 91 CB   STA  PLAYFLAG ;Flag für PLAY rücksetzen
 826A  A9 00      LDA  #$00     ;Interrupt-Maske im VIC
 826C  8D 1A D0   STA  VICIRQM  ;löschen (Int. verhindern)
 826F  58         CLI
 8270  AD 18 D0   LDA  VICADS   ;Adreß-Steuer-Byte des VIC
 8273  C9 0B      CMP  #$0B     ;$0B ist Wert für HIRES
 8275  D0 06      BNE  $827D
 8277  20 EC 82   JSR  BASROMAU
 827A  20 37 A8   JSR  NRM      ;Standard-Werte herstellen
 827D  20 F3 82   JSR  BASROMEI
 8280  AD B2 C5   LDA  AUTOFLAG ;AUTO-Modus
 8283  C9 0A      CMP  #$0A     ;$0A = aktiv
 8285  F0 03      BEQ  $828A    ;ja, dann weiter
 8287  4C 83 A4   JMP  $A483    ;sonst zu Basic-Schleife
```

```
828A  18          CLC
828B  AD FE 01    LDA  AUTOZI      ;momentane Zeilennummer
828E  6D B1 C5    ADC  AUTOINC     ;plus Inkrement
8291  85 63       STA  $63         ;für Umwandlung
8293  8D FE 01    STA  AUTOZI      ;und als neue Nummer sp.
8296  AD FF 01    LDA  AUTOZI+1    ;Übertrag
8299  69 00       ADC  #$00        ;addieren
829B  85 62       STA  $62         ;und
829D  8D FF 01    STA  AUTOZI+1    ;speichern
82A0  A2 90       LDX  #$90        ;Exponent ist $90
82A2  38          SEC
82A3  20 49 BC    JSR  $BC49       ;nach Fließkomma wandeln
82A6  20 DF BD    JSR  $BDDF       ;in Text (ab $0100) umw.
82A9  A2 00       LDX  #$00        ;als Zeiger in Text
82AB  BD 00 01    LDA  $0100,X     ;Textzeichen holen
82AE  F0 06       BEQ  $82B6       ;Ende bei $00
82B0  9D 77 02    STA  TASTPUFF,X ;in Tastat.puff. ablegen
82B3  E8          INX              ;nächstes Zeichen
82B4  D0 F5       BNE  $82AB       ;unbedingter Sprung
82B6  A9 20       LDA  #$20        ;Leerzeichen
82B8  9D 77 02    STA  TASTPUFF,X ; in Tastaturpuffer
82BB  E8          INX              ;Pufferzeiger eröhen
82BC  86 C6       STX  $C6         ;und speichern
82BE  A9 00       LDA  #$00
82C0  8D B2 C5    STA  AUTOFLAG    ;Auto-Modus rücksetzen
82C3  20 60 A5    JSR  $A560       ;Eine Zeile einlesen
82C6  84 AA       STY  $AA         ;Y sichern
82C8  A0 00       LDY  #$00
82CA  C8          INY
82CB  B9 00 02    LDA  $0200,Y     ;Puffer durchsuchen
82CE  F0 12       BEQ  $82E2       ;Zeilenende ?
82D0  C9 3A       CMP  #$3A        ;$39 ist Code für '9'
82D2  90 F6       BCC  $82CA       ;kleiner= '9',dann weiter
82D4  C9 5B       CMP  #$5B        ;$5A ist Code für 'z'
82D6  B0 F2       BCS  $82CA       ;größer 'z', dann weiter
82D8  A4 AA       LDY  $AA         ;Y wieder herstellen
82DA  A9 0A       LDA  #$0A        ;Auto-Modus wieder ein
82DC  8D B2 C5    STA  AUTOFLAG
82DF  4C 86 A4    JMP  $A486       ;zur Basic-Eingabeschleife

82E2  A9 00       LDA  #$00        ;Auto-Modus aus
82E4  8D B2 C5    STA  AUTOFLAG
82E7  A4 AA       LDY  $AA         ;Y wieder herstellen
82E9  4C 86 A4    JMP  $A486       ;zur Basic-Eingabeschleife

BASROMAU:
82EC  A5 01       LDA  $01         ;Hole Prozessorport
82EE  29 FE       AND  #$FE        ;lösche Bit 0 (LORAM)
82F0  85 01       STA  $01         ;Speichere Prozessorport
82F2  60          RTS
```

```
BASROMEI:
 82F3  A5 01      LDA  $01        ;Hole Prozessorport
 82F5  09 01      ORA  #$01       ;sezte Bit 0
 82F7  85 01      STA  $01        ;Speichere Port
 82F9  60         RTS

SNMI:
 82FA  48         PHA             ;Register A,X,Y retten
 82FB  8A         TXA
 82FC  48         PHA
 82FD  98         TYA
 82FE  48         PHA
 82FF  20 0E 94   JSR  KERROMEIN  ;KERNAL ein
 8302  A9 7F      LDA  #$7F       ;alle Bits im Maskenreg.
 8304  8D 0D DD   STA  CIA2ICR    ;des CIA2 löschen
 8307  AC 0D DD   LDY  CIA2ICR    ;Int.-Register lesen
 830A  30 35      BMI  $8341      ;wenn NMI von CIA2, dann
                                  ;zu $8341
 830C  20 BC F6   JSR  $F6BC      ;STOP-Taste abfragen
 830F  20 E1 FF   JSR  $FFE1      ;STOP-Taste prüfen
 8312  D0 2D      BNE  $8341      ;wenn nicht, dann weiter

SBRK:                             ;wird bei STOP/RESTORE und
                                  ;bei BRK angesprungen
 8314  A9 04      LDA  #$04
 8316  8D 88 02   STA  VIDRAMHI   ;Video-RAM ab $0400
 8319  20 A3 FD   JSR  $FDA3      ;Interrupts vorbereiten
 831C  20 18 E5   JSR  $E518      ;Bildschirm initialisieren
 831F  A9 0F      LDA  #$0F       ;hellgrau
 8321  8D 21 D0   STA  VICHIFAR   ;als Hintergrundfarbe
 8324  A9 06      LDA  #$06       ;blau
 8326  8D 20 D0   STA  VICRAFAR   ;als Rahmenfarbe
 8329  A9 00      LDA  #$00       ;schwarz
 832B  8D 86 02   STA  AKTFARB    ;als aktuelle Zeichenfarbe
 832E  85 D4      STA  $D4        ;Hochkommamodus rücksetzen
 8330  8D B4 C5   STA  MOBBEW     ;Spritebewegung aus
 8333  8D B3 C5   STA  GMEMFLAG   ;Speicherverteilung norm.
 8336  A9 08      LDA  #$08       ;Code f. (Sh/C= verh.)
 8338  8D 91 CB   STA  PLAYFLAG   ;Flag für PLAY rücksetzen
 833B  20 D2 FF   JSR  BSOUT      ;Zeichen CHR$(8) ausgeben
 833E  6C 02 A0   JMP  (BNMIVEC);zur normalen NMI-Behandl.

 8341  4C 72 FE   JMP  $FE72      ;zur Beh. von RS-232-NMI

BEFMERGE:
 8344  20 DB 83   JSR  INCBASBZ   ;Code überlesen
 8347  20 F3 82   JSR  BASROMEI   ;Basic ein
 834A  A9 00      LDA  #$00
 834C  85 0A      STA  $0A        ;LOAD/VERIFY-Flag
 834E  20 D4 E1   JSR  $E1D4      ;Parameter f. LOAD holen
 8351  38         SEC
```

```
8352  A5 2D      LDA  $2D       ;Basic-Programm-Ende
8354  E9 02      SBC  #$02      ;um 2 vermindern
8356  AA         TAX            ;und als Startadresse
8357  A5 2E      LDA  $2E       ;an LOAD-Routine
8359  E9 00      SBC  #$00      ;übergeben
835B  A8         TAY            ;(X/Y)=(Low/High)
835C  A9 00      LDA  #$00      ;LOAD/VERIFY-Flag
835E  20 D5 FF   JSR  $FFD5     ;LOAD-Routine
8361  B0 19      BCS  $837C     ;Error,dann Fehlerbehandl.
8363  20 B7 FF   JSR  $FFB7     ;READST; Status holen
8366  29 BF      AND  #$BF      ;Bit 6 ignorieren
8368  F0 05      BEQ  $836F     ;ohne Fehler, dann weiter
836A  A2 1D      LDX  #$1D      ;Nummer von 'load error'
836C  4C 37 A4   JMP  $A437     ;Fehlermeldung ausgeben

836F  86 2D      STX  $2D       ;Endadresse als Basic-
8371  84 2E      STY  $2E       ;Programm-Ende speichern
8373  20 59 A6   JSR  $A659     ;Prgz. auf Anf.; CLR
8376  20 33 A5   JSR  $A533     ;Programmzeilen neu binden
8379  4C 74 A4   JMP  $A474     ;ready.

837C  4C F9 E0   JMP  KERERRB   ;zur KERNAL-Fehlerbehandl.

BEFRENUM:
837F  A9 01      LDA  #$01      ;Vektor ($A8,$A9) auf
8381  85 A8      STA  $A8       ;$0801 (Basic-Anfang)
8383  A9 08      LDA  #$08      ;richten
8385  85 A9      STA  $A9
8387  20 DB 83   JSR  INCBASBZ  ;Renumber-Code überlesen
838A  20 27 94   JSR  SGETADR   ;Adresswert (Beginn) holen
838D  84 A4      STY  $A4       ;und nach $A4,$A5 bringen
838F  85 A5      STA  $A5
8391  20 31 80   JSR  SCHKCOM   ;Komma ?
8394  20 27 94   JSR  SGETADR   ;Wert (Schrittweite) holen
8397  8C AC C5   STY  AY        ;und ab $C5AC ablegen
839A  8D AD C5   STA  AY+1
839D  A0 00      LDY  #$00
839F  B1 A8      LDA  ($A8),Y   ;Vorwärtszeiger Low
83A1  85 09      STA  $09       ;nach $09
83A3  20 F0 9A   JSR  INCA8A9   ;nächst. Byte im Basic-Pr.
83A6  B1 A8      LDA  ($A8),Y   ;Vorwärtszeiger High
83A8  85 0A      STA  $0A       ;nach $0A
83AA  20 F0 9A   JSR  INCA8A9   ;nächst. Byte im Basic-Pr.
83AD  A5 A4      LDA  $A4       ;Soll-Zeilennr. Low
83AF  91 A8      STA  ($A8),Y   ;als Zeilennr. Low speich.
83B1  20 F0 9A   JSR  INCA8A9   ;nächst. Byte im Basic-Pr.
83B4  A5 A5      LDA  $A5       ;Soll-Zeilennr. High
83B6  91 A8      STA  ($A8),Y   ;als Zeilennr. High spei.
83B8  A5 09      LDA  $09       ;Vorwärtszeiger Low
83BA  05 0A      ORA  $0A       ;und High
83BC  F0 1A      BEQ  $83D8     ;beide Null, dann fertig
```

```
 83BE   A5 09         LDA  $09          ;Vorwärtszeiger nach
 83C0   85 A8         STA  $A8          ;$A8,$A9 bringen
 83C2   A5 0A         LDA  $0A
 83C4   85 A9         STA  $A9
 83C6   18            CLC
 83C7   A5 A4         LDA  $A4          ;Soll-Zeilennummer
 83C9   6D AC C5      ADC  AY           ;um Inkrement erhöhen
 83CC   85 A4         STA  $A4
 83CE   A5 A5         LDA  $A5          ;High-Byte
 83D0   6D AD C5      ADC  AY+1
 83D3   85 A5         STA  $A5
 83D5   4C 9F 83      JMP  $839F        ;nächste Zeile umnumerier.

 83D8   4C 2A 82      JMP  ENDSMB       ;Befehl abschließen

INCBASBZ:
 83DB   E6 7A         INC  $7A          ;erhöhe Low-Byte
 83DD   D0 02         BNE  $83E1        ;wenn kein Übertrag,weiter
 83DF   E6 7B         INC  $7B          ;sonst erhöhe High-Byte
 83E1   60            RTS

TABBEF:                                 ;Tabelle der Befehlswörter
                                        ;von Simon's Basic;
                                        ;Trennung voneinaner durch
                                        ;Klammeraffe (§); Lfd. Nr.
83E2   48 49 52 45 53 40                    hires§         1
83E8   50 4C 4F 54 40                       plot§          2
83ED   4C 49 4E 45 40                       line§          3
83F2   42 4C 4F 43 4B 40                    block§         4
83F8   46 43 48 52 40                       fchr§          5
83FD   46 43 4F 4C 40                       fcol§          6
8402   46 49 4C 4C 40                       fill§          7
8407   52 45 43 40                          rec§           8
840B   52 4F 54 40                          rot§           9
840F   44 52 41 57 40                       draw§         10
8414   43 48 41 52 40                       char§         11
8419   48 49 20 43 4F 4C 40                 hi col§       12
8420   49 4E 56 40                          inv§          13
8424   46 52 41 43 40                       frac§         14
8429   4D 4F 56 45 40                       move§         15
842E   50 4C 41 43 45 40                    place§        16
8434   55 50 42 40                          upb§          17
8438   55 50 57 40                          upw§          18
843C   4C 45 46 54 57 40                    leftw§        19
8442   4C 45 46 54 42 40                    leftb§        20
8448   44 4F 57 4E 42 40                    downb§        21
844E   44 4F 57 4E 57 40                    downw§        22
8454   52 49 47 48 54 42 40                 rightb§       23
845B   52 49 47 48 54 57 40                 rightw§       24
8462   4D 55 4C 54 49 40                    multi§        25
8468   43 4F 4C 4F 55 52 40                 colour§       26
```

846F	4D 4D 4F 42 40	mmob§	27
8474	42 46 4C 41 53 48 40	bflash§	28
847B	4D 4F 42 20 53 45 54 40	mob set§	29
8483	4D 55 53 49 43 40	music§	30
8489	46 4C 41 53 48 40	flash§	
848F	52 45 50 45 41 54 40	repeat§	
8496	50 4C 41 59 40	play§	
849B	3E 3E 40	µµ§	
849E	43 45 4E 54 52 45 40	centre§	35
84A5	45 4E 56 45 4C 4F 50 45 40	envelope§	
84AE	43 47 4F 54 4F 40	cgoto§	
84B4	57 41 56 45 40	wave§	
84B9	46 45 54 43 48 40	fetch§	
84BF	41 54 28 40	at(§	40
84C3	55 4E 54 49 4C 40	until§	
84C9	3E 3E 40	µµ§	
84CC	3E 3E 40	µµ§	
84CF	55 53 45 40	use§	
84D3	3E 3E 40	µµ§	45
84D6	47 4C 4F 42 41 4C 40	global§	
84DD	3E 3E 40	µµ§	
84E0	52 45 53 45 54 40	reset§	
84E6	50 52 4F 43 40	proc§	
84EB	43 41 4C 4C 40	call§	50
84F0	45 58 45 43 40	exec§	
84F5	45 4E 44 20 50 52 4F 43 40	end proc§	
84FE	45 58 49 54 40	exit§	
8503	45 4E 44 20 4C 4F 4F 50 40	end loop§	
850C	4F 4E 20 4B 45 59 40	on key§	55
8513	44 49 53 41 42 4C 45 40	disable§	
851B	52 45 53 55 4D 45 40	resume§	
8522	4C 4F 4F 50 40	loop§	
8527	44 45 4C 41 59 40	delay§	
852D	3E 3E 40	µµ§	60
8530	3E 3E 40	µµ§	
8533	3E 3E 40	µµ§	
8536	3E 3E 40	µµ§	
8539	53 45 43 55 52 45 40	secure§	
8540	44 49 53 41 50 41 40	disapa§	65
8547	43 49 52 43 4C 45 40	circle§	
854E	4F 4E 20 45 52 52 4F 52 40	on error§	
8557	4E 4F 20 45 52 52 4F 52 40	no error§	
8560	4C 4F 43 41 4C 40	local§	
8566	52 43 4F 4D 50 40	rcomp§	70
856C	45 4C 53 45 40	else§	
8571	52 45 54 52 41 43 45 40	retrace§	
8579	54 52 41 43 45 40	trace§	
857F	44 49 52 40	dir§	
8583	50 41 47 45 40	page§	75
8588	44 55 4D 50 40	dump§	
858D	46 49 4E 44 40	find§	

```
8592  4F 50 54 49 4F 4E 40           option§
8599  41 55 54 4F 40                 auto§
859E  4F 4C 44 40                    old§           80
85A2  4A 4F 59 40                    joy§
85A6  4D 4F 44 40                    mod§
85AA  44 49 56 40                    div§
85AE  3E 3E 40                       µµ§
85B1  44 55 50 40                    dup§           85
85B5  49 4E 4B 45 59 40              inkey§
85BB  49 4E 53 54 40                 inst§
85C0  54 45 53 54 40                 test§
85C5  4C 49 4E 40                    lin§
85C9  45 58 4F 52 40                 exor§          90
85CE  49 4E 53 45 52 54 40           insert§
85D5  50 4F 54 40                    pot§
85D9  50 45 4E 58 40                 penx§
85DE  3E 3E 40                       µµ§
85E1  50 45 4E 59 40                 peny§          95
85E6  53 4F 55 4E 44 40              sound§
85EC  47 52 41 50 48 49 43 53 40     graphics§
85F5  44 45 53 49 47 4E 40           design§
85FC  52 4C 4F 43 4D 4F 42 40        rlocmob§
8604  43 4D 4F 42 40                 cmob§          100
8609  42 43 4B 47 4E 44 53 40        bckgnds§
8611  50 41 55 53 45 40              pause§
8617  4E 52 4D 40                    nrm§
861B  4D 4F 42 20 4F 46 46 40        mob off§
8623  4F 46 46 40                    off§           105
8627  41 4E 47 4C 40                 angl§
862C  41 52 43 40                    arc§
8630  43 4F 4C 44 40                 cold§
8635  53 43 52 53 56 40              scrsv§
863B  53 43 52 4C 44 40              scrld§         110
8641  54 45 58 54 40                 text§
8646  43 53 45 54 40                 cset§
864B  56 4F 4C 40                    vol§
864F  44 49 53 4B 40                 disk§
8654  48 52 44 43 50 59 40           hrdcpy§        115
865B  4B 45 59 40                    key§
865F  50 41 49 4E 54 40              paint§
8665  4C 4F 57 20 43 4F 4C 40        low col§
866D  43 4F 50 59 40                 copy§
8672  4D 45 52 47 45 40              merge§         120
8678  52 45 4E 55 4D 42 45 52 40     renumber§
8681  4D 45 4D 40                    mem§
8685  44 45 54 45 43 54 40           detect§
868C  43 48 45 43 4B 40              check§
8692  44 49 53 50 4C 41 59 40        display§       125
869A  45 52 52 40                    err§
869E  4F 55 54 40                    out§           127
86A2  00
```

```
 86A3  44 53 2D 43 42 4D 00     "ds-cbm"

TAB?:
 86AB  20 80 FF 2E 30 00 80 FF 30 2E     (???)

SGETSTRN:
 86B4  20 DB 83   JSR  INCBASBZ ;überlies 1 Zeichen
SGETSTR:
 86B7  20 3A 80   JSR  SFRMEVL  ;hole Ausdruck
 86BA  20 43 80   JSR  SFRESTR  ;hole Stringparameter
 86BD  85 69      STA  $69      ;Länge nach $69
 86BF  A5 23      LDA  $23      ;Adresse nach $22,$23
 86C1  A6 22      LDX  $22
 86C3  60         RTS

****  Sprungtabelle für Befehlsausführung
****  Zieladresse ist um 1 erhöht, da Aufruf mit RTS

;                 Zieladr.   Zielmarke      Lfd.Nr.
SPRTAB:
 86C4  FE 91      $91FF      befhires          1
 86C6  66 92      $9267      befplot           2
 86C8  4F 93      $9350      befline           3
 86CA  6D BC      $BC6E      befblock          4
 86CC  CF A2      $A2D0      beffchr           5
 86CE  E1 A2      $A2E2      beffcol           6
 86D0  96 A2      $A297      beffill           7
 86D2  F6 BA      $BAF7      befrec            8
 86D4  5D A1      $A15E      befrot            9
 86D6  56 A0      $A057      befdraw          10
 86D8  85 A1      $A186      befchar          11
 86DA  6F A2      $A270      befhicol         12
 86DC  5C AC      $AC5D      movebef          13
 86DE  89 88      $888A      badmode          14
 86E0  5C AC      $AC5D      movebef          15
 86E2  89 88      $888A      badmode          16
 86E4  5C AC      $AC5D      movebef          17
 86E6  5C AC      $AC5D      movebef          18
 86E8  5C AC      $AC5D      movebef          19
 86EA  5C AC      $AC5D      movebef          20
 86EC  5C AC      $AC5D      movebef          21
 86EE  5C AC      $AC5D      movebef          22
 86F0  5C AC      $AC5D      movebef          23
 86F2  5C AC      $AC5D      movebef          24
 86F4  E9 94      $94EA      befmulti         25
 86F6  36 95      $9537      befcolour        26
 86F8  C7 95      $95C8      befmmob          27
 86FA  68 96      $9669      befbflash        28
 86FC  D9 96      $96DA      befmobset        29
 86FE  58 97      $9759      befmusic         30
 8700  8F 97      $9790      befflash         31
```

8702	F6 9A	$9AF7	befrepeat	32
8704	17 99	$9918	befplay	33
8706	17 99	$9918	befplay	34
8708	46 99	$9947	befcentre	35
870A	78 99	$9979	befenvelope	36
870C	E7 99	$99E8	befcgoto	37
870E	F6 99	$99F7	befwave	38
8710	03 B0	$B004	beffetch	39
8712	89 88	$888A	badmode	40
8714	14 9B	$9B15	befuntil	41
8716	3B A3	$A33C	befglobal	42
8718	3B A3	$A33C	befglobal	43
871A	92 B3	$B393	befuse	44
871C	92 B3	$B393	befuse	45
871E	3B A3	$A33C	befglobal	46
8720	04 9C	$9C05	befreset	47
8722	04 9C	$9C05	befreset	48
8724	2F 9F	$9F30	befproc	49
8726	29 9C	$9C2A	befcall	50
8728	E2 9C	$9CE3	befexec	51
872A	18 9D	$9D19	befendproc	52
872C	36 9D	$9D37	befexit	53
872E	6E 9D	$9D6F	befendloop	54
8730	88 9D	$9D89	befonkey	55
8732	B1 9D	$9DB2	befdisable	56
8734	0E 9E	$9E0F	befresume	57
8736	23 9E	$9E24	befloop	58
8738	41 9E	$9E42	befdelay	59
873A	4A 9E	$9E4B	befsecure	60
873C	4A 9E	$9E4B	befsecure	61
873E	4A 9E	$9E4B	befsecure	62
8740	4A 9E	$9E4B	befsecure	63
8742	4A 9E	$9E4B	befsecure	64
8744	73 91	$9174	bef0	65
8746	88 94	$9489	befcircle	66
8748	6D 9E	$9E6E	befonerr	67
874A	93 9E	$9E94	befnoerr	68
874C	F3 A2	$A2F4	beflocal	69
874E	1E 9F	$9F1F	befrcomp	70
8750	2F 9F	$9F30	befproc	71
8752	47 9B	$9B48	befretrace	72
8754	59 9B	$9B5A	beftrace	73
8756	45 95	$9546	befdir	74
8758	35 98	$9836	befpage	75
875A	3E 9F	$9F3F	befdump	76
875C	93 A5	$A594	beffind	77
875E	62 9B	$9B63	befoption	78
8760	D5 9B	$9BD6	befauto	79
8762	D9 9E	$9EDA	befold	80
8764	89 88	$888A	badmode	81
8766	89 88	$888A	badmode	82

8768	89 88	$888A	badmode	83
876A	89 88	$888A	badmode	84
876C	89 88	$888A	badmode	85
876E	89 88	$888A	badmode	86
8770	89 88	$888A	badmode	87
8772	89 88	$888A	badmode	88
8774	89 88	$888A	badmode	89
8776	89 88	$888A	badmode	90
8778	89 88	$888A	badmode	91
877A	89 88	$888A	badmode	92
877C	89 88	$888A	badmode	93
877E	89 88	$888A	badmode	94
8780	89 88	$888A	badmode	95
8782	89 88	$888A	badmode	96
8784	89 88	$888A	badmode	97
8786	2D A6	$A62E	befdesign	98
8788	67 A7	$A768	befrlocm	99
878A	A5 A7	$A7A6	befcmob	100
878C	B4 A7	$A7B5	befbckgnds	101
878E	E1 A7	$A7E2	befpause	102
8790	30 A8	$A831	befnrm	103
8792	55 A8	$A856	befmoboff	104
8794	64 A8	$A865	befoff	105
8796	0F AB	$AB10	befangl	106
8798	75 A8	$A876	befarc	107
879A	46 81	$8147	befcold	108
879C	8E B1	$B18F	befscrsv	109
879E	DB B1	$B1DC	befscrld	110
87A0	28 B2	$B229	beftext	111
87A2	0C B3	$B30D	befcset	112
87A4	3E B3	$B33F	befvol	113
87A6	5C B3	$B35D	befdisk	114
87A8	9F B4	$B4A0	befhrdcpy	115
87AA	8F B5	$B590	befkey	116
87AC	E7 B5	$B5E8	befpaint	117
87AE	DB 93	$93DC	beflowcol	118
87B0	F0 B9	$B9F1	befcopy	119
87B2	43 83	$8344	befmerge	120
87B4	7E 83	$837F	befrenum	121
87B6	AC BD	$BDAD	befmem	122
87B8	F6 BD	$BDF7	befdetect	123
87BA	F6 BD	$BDF7	befdetect	124
87BC	12 BE	$BE13	befdisplay	125
87BE	89 88	$888A	badmode	126
87C0	D0 9E	$9ED1	befout	127

```
FEHLERMELDUNGEN:
 87c2   50 52 4f 43 20 4e 4f 54 20    proc not found
 87cb   46 4f 55 4e 44 00
 87d1   49 4e 53 45 52 54 20 54 4f    insert too large
 87da   4f 20 4c 41 52 47 45 00
 87e2   53 54 52 49 4e 47 20 54 4f    string to large
 87eb   20 4c 41 52 47 45 00
 87f2   4e 4f 54 20 42 49 4e 41 52    not binary char
 87fb   59 20 43 48 41 52 00
 8802   4e 4f 54 20 48 45 58 20 43    not hex char
 880b   48 41 52 00
 880f   45 4e 44 20 50 52 4f 43 20    end proc without exec
 8818   57 49 54 48 4f 55 54 20 45
 8821   58 45 43 00
 8825   45 4e 44 20 4c 4f 4f 50 20    end loop without loop
 882e   57 49 54 48 4f 55 54 20 4c
 8837   4f 4f 50 00
 883b   55 4e 54 49 4c 20 57 49 54    until without repeat
 8844   48 4f 55 54 20 52 45 50 45
 884d   41 54 00
 8850   53 54 41 43 4b 20 54 4f 4f    stack too large
 8859   20 4c 41 52 47 45 00
 8860   54 4f 4f 20 46 45 57 20 4c    too few lines
 8869   49 4e 45 53 00
 886e   42 41 44 20 43 48 41 52 20    bad char for a mob
 8877   46 4f 52 20 41 20 4d 4f 42
 8880   00
 8881   42 41 44 20 4d 4f 44 45 00    bad mode

BADMODE:
 888A   A9 00      LDA  #$00       ;Fehler Nr. 0

SERROUT:                           ;gibt Fehlermeldung (Feh-
                                   ;lernummer im Akku) aus
 888C   0A         ASL  A          ;Fehlernummer verdoppeln
 888D   A8         TAY             ;als Zeiger in Fehlm.tab.
 888E   B9 BB 88   LDA  $FEHLMTAB+1,Y ;Adresse der Fehler-
 8891   85 21      STA  $21           ;meldung nach $20,$21
 8893   B9 BA 88   LDA  $FEHLMTAB,Y
 8896   85 20      STA  $20
 8898   20 F3 82   JSR  BASROMEI   ;Basic ein
 889B   20 0E 94   JSR  KERROMEIN  ;KERNAL ein
 889E   A9 3F      LDA  #$3F       ;Fragezeichen
 88A0   20 D2 FF   JSR  BSOUT      ;ausgeben
 88A3   A0 00      LDY  #$00
 88A5   B1 20      LDA  ($20),Y    ;Zeichen der Fehlerm.
 88A7   F0 06      BEQ  $88AF      ;Endekriterium = 0
 88A9   20 D2 FF   JSR  BSOUT      ;Zeichen ausgeben
 88AC   C8         INY             ;nächstes Zeichen
 88AD   D0 F6      BNE  $88A5      ;Schleife
```

```
 88AF   A4 3A       LDY   $3A        ;Zeilennummer high
 88B1   C8          INY              ;wenn ($3A) = $FF
 88B2   F0 03       BEQ   $88B7      ;war, dann weiter
 88B4   20 C2 BD    JSR   $BDC2      ;'in Zeilennr.' ausgeben
 88B7   4C 86 E3    JMP   WARM       ;zum Basic-Warmstart

FEHLMTAB:        ; Adresse der Fehlermeldung     Lfd.Nr.
 88BA   81 88        $8881                          0
 88BC   C2 87        $87C2                          1
 88BE   D1 87        $87D1                          2
 88C0   E2 87        $87E2                          3
 88C2   F2 87        $87F2                          4
 88C4   02 88        $8802                          5
 88C6   0F 88        $880F                          6
 88C8   25 88        $8825                          7
 88CA   3B 88        $883B                          8
 88CC   50 88        $8850                          9
 88CE   60 88        $8860                         10
 88D0   6E 88        $886E                         11
 88D2   81 88        $8881                         12

FNJOY:
 88D4   AD 00 DC    LDA   CIA1PRA    ;CIA1 Port A
 88D7   85 20       STA   $20        ;zwischenspeichern
 88D9   29 EF       AND   #$EF       ;Bit 4 (Feuer) wegblenden
 88DB   8D 70 CB    STA   JOYWERT    ;speichern
 88DE   A2 00       LDX   #$00
 88E0   BD 12 89    LDA   TABJOY,X   ;mit Wert aus Tabelle
 88E3   CD 70 CB    CMP   JOYWERT    ;vergleichen
 88E6   F0 0D       BEQ   $88F5      ;gefunden, dann weiter
 88E8   E8          INX              ;nächsten Tabellenwert
 88E9   E0 08       CPX   #$08       ;Tabellende nach 8 Werten
 88EB   D0 F3       BNE   $88E0      ;nicht err., dann weiter
 88ED   A9 00       LDA   #$00       ;nicht gef., dann 0
 88EF   8D 70 CB    STA   JOYWERT    ;speichern
 88F2   4C F9 88    JMP   $88F9

 88F5   E8          INX              ;Tabellenindex + 1
 88F6   8E 70 CB    STX   JOYWERT    ;als Wert speichern
 88F9   A5 20       LDA   $20        ;urspr. Abfragewert
 88FB   29 90       AND   #$90       ;Bit 7 und Bit 4 checken
 88FD   D0 08       BNE   $8907      ;gesetzt, dann weiter
 88FF   AD 70 CB    LDA   JOYWERT    ;Wert mit
 8902   09 80       ORA   #$80       ;128 oder-verknüpfen
 8904   8D 70 CB    STA   JOYWERT
 8907   AC 70 CB    LDY   JOYWERT    ;Low-Byte  v. Ergebnis
 890A   A2 00       LDX   #$00       ;High-Byte v. Ergebnis
 890C   20 DB 83    JSR   INCBASBZ   ;Code überlesen
 890F   4C CD 8C    JMP   SXYFLP     ;nach Fließk. umwandeln
```

```
TABJOY:
 8912   6E 66 67 65 6D 69 6B 6A  ;Werte f Joyst.-Stellungen

FNPOT:
 891A   20 73 00   JSR   CHRGET    ;Code überlesen
 891D   20 FA AE   JSR   CHKKLA
 8920   20 15 94   JSR   SGETADR2
 8923   20 F7 AE   JSR   CHKKLZ
 8926   A5 65      LDA   $65       ;FAC-Exp.
 8928   D0 06      BNE   $8930     ;wenn nicht 0, dann weiter
 892A   AC 19 D4   LDY   $D419     ;Paddle-Wert 0 vom SID
 892D   4C 33 89   JMP   $8933

 8930   AC 1A D4   LDY   $D41A     ;Paddle-Wert 1 vom SID
 8933   84 63      STY   $63       ;Low-Byte Ergebnis
 8935   A2 00      LDX   #$00      ;High-Byte
 8937   4C CD 8C   JMP   SXYFLP    ;nach Fließk. wandeln

FNPENX:
 893A   20 4E 89   JSR   PENABFR   ;Abfrage-Routine (s.u.)
 893D   AC 00 C5   LDY   LPX
 8940   4C 46 89   JMP   $8946     ;zur Umwandlung n. Fließk.

FNPENY:
 8943   AC 01 C5   LDY   LPY
 8946   A2 00      LDX   #$00
 8948   20 DB 83   JSR   INCBASBZ   ;Code überlesen
 894B   4C CD 8C   JMP   SXYFLP     ;X/Y nach Fließkomma

PENABFR:
 894E   AD 19 D0   LDA   VICIRQF    ;IRQ-Flags im VIC
 8951   29 08      AND   #$08       ;Light-Pen-Strobe ?
 8953   F0 F9      BEQ   PENABFR    ;wenn nicht, dann nochmal
 8955   AD 13 D0   LDA   VICLPX     ;LPX-Wert vom VIC
 8958   8D 00 C5   STA   LPX        ;speichern
 895B   AD 14 D0   LDA   VICLPY     ;analog f. Y
 895E   8D 01 C5   STA   LPY
 8961   AD 19 D0   LDA   VICIRQF    ;IRQ-Flags im VIC
 8964   29 F7      AND   #$F7       ;Bit 3 löschen
 8966   8D 19 D0   STA   VICIRQF    ;Flags speichern
 8969   60         RTS

TRACE1:
 896A   A5 39      LDA   $39        ;akt. Zeilennr.
 896C   85 63      STA   $63        ;nach $63,$62 bringen
 896E   A5 3A      LDA   $3A
 8970   85 62      STA   $62
 8972   A5 7B      LDA   $7B        ;akt. Befehlszeiger
 8974   C9 03      CMP   #$03       ;Test auf Direktmodus
 8976   B0 01      BCS   $8979      ;nein, dann weiter
 8978   60         RTS
```

```
8979  A2 90     LDX  #$90        ;Exponent f. SXFLP
897B  38        SEC
897C  20 88 80  JSR  SXFLP       ;Zeilennr. im FAC
897F  20 B5 80  JSR  SFACASC     ;in Text umwandeln
8982  20 98 89  JSR  TRSHIFT     ;alte Zeilennrn. verschieb
8985  20 3A 8A  JSR  WEISSECK    ;weißes Eck am Bildschirm
8988  A2 00     LDX  #$00
898A  BD 00 01  LDA  $0100,X     ;Text ab $0100 (='FAC')
898D  F0 08     BEQ  $8997       ;Ende bei 0-Kode
898F  49 80     EOR  #$80        ;Bit 7 (RVS) setzen
8991  9D 55 C5  STA  TRACETAB+36,X ;an Tabelle anfügen
8994  E8        INX              ;nächstes Zeichen
8995  D0 F3     BNE  $898A       ;unbed. Sprung
8997  60        RTS

TRSHIFT:
8998  78        SEI
8999  A9 C5     LDA  #$C5        ;Zeiger $23,24 auf $C531
899B  85 24     STA  $24
899D  A9 31     LDA  #$31
899F  85 23     STA  $23
89A1  A9 C5     LDA  #$C5        ;Zeiger $20,21 auf $C537
89A3  85 21     STA  $21
89A5  A9 37     LDA  #$37
89A7  85 20     STA  $20
89A9  A0 00     LDY  #$00        ;Zähler in einem Eintrag
89AB  A2 00     LDX  #$00        ;Zähler für Einträge
89AD  B1 20     LDA  ($20),Y     ;1 Byte transferieren
89AF  91 23     STA  ($23),Y
89B1  C8        INY              ;nächstes Byte des Eintr.
89B2  C0 05     CPY  #$05        ;1 Eintrag = 6 Byte
89B4  D0 F7     BNE  $89AD       ;'innere' Schleife
89B6  18        CLC
89B7  A5 20     LDA  $20         ;$20,$21 um 6 erhöhen
89B9  69 06     ADC  #$06
89BB  85 20     STA  $20
89BD  A5 21     LDA  $21
89BF  69 00     ADC  #$00
89C1  85 21     STA  $21
89C3  18        CLC
89C4  A5 23     LDA  $23         ;$23,$24 um 6 erhöhen
89C6  69 06     ADC  #$06
89C8  85 23     STA  $23
89CA  A5 24     LDA  $24
89CC  69 00     ADC  #$00
89CE  85 24     STA  $24
89D0  E8        INX              ;nächsten Eintrag
89D1  A0 00     LDY  #$00
89D3  E0 06     CPX  #$06        ;7 Einträge
89D5  D0 D6     BNE  $89AD       ;'äußere' Schleife
89D7  A9 A0     LDA  #$A0        ;(RVS-Leerzeichen)
```

```
 89D9   91 23       STA  ($23),Y  ;als letzten Eintrag spei.
 89DB   C8          INY           ;nächstes Zeichen
 89DC   C0 05       CPY  #$05     ;6 Zeichen
 89DE   D0 F9       BNE  $89D9    ;Schleife
 89E0   58          CLI
 89E1   60          RTS

TRACESHOW:
 89E2   A5 7B       LDA  $7B      ;Programmzähler
 89E4   C9 03       CMP  #$03     ;Direkt-Modus
 89E6   B0 01       BCS  TRACESH1 ;nein, dann weiter
 89E8   60          RTS

TRACESH1:
 89E9   20 3A 8A    JSR  WEISSECK ;weißes Eck am Bildschirm
 89EC   78          SEI
 89ED   A9 C5       LDA  #$C5     ;Zeiger $23,$24 auf $C531
 89EF   85 24       STA  $24
 89F1   A9 31       LDA  #$31
 89F3   85 23       STA  $23
 89F5   A9 04       LDA  #$04     ;Zeiger $20,$21 auf $0422
 89F7   85 21       STA  $21      ;(Video-Ram rechts oben)
 89F9   A9 22       LDA  #$22
 89FB   85 20       STA  $20
 89FD   A0 00       LDY  #$00
 89FF   A2 00       LDX  #$00
 8A01   A9 A3       LDA  #$A3     ;(RVS-#)
 8A03   91 20       STA  ($20),Y  ;als 1.Zeichen schreiben
 8A05   B1 23       LDA  ($23),Y  ;1. Zeichen aus Tabelle
 8A07   C8          INY
 8A08   91 20       STA  ($20),Y  ;an nächste Bildsch.-Pos.
 8A0A   C0 05       CPY  #$05     ;1 Eintrag = 6 Zeichen
 8A0C   D0 F7       BNE  $8A05    ;'innere' Schleife
 8A0E   20 CE 94    JSR  20PL40   ;Zeiger $20,$21 um 40 erh.
 8A11   18          CLC
 8A12   A5 23       LDA  $23      ;Zeiger $23,$24 um 6 erh.
 8A14   69 06       ADC  #$06
 8A16   85 23       STA  $23
 8A18   A5 24       LDA  $24
 8A1A   69 00       ADC  #$00
 8A1C   85 24       STA  $24
 8A1E   E8          INX             ;nächsten Eintrag
 8A1F   A0 00       LDY  #$00
 8A21   E0 06       CPX  #$06       ;7 Einträge
 8A23   D0 DC       BNE  $8A01      ;'äußere' Schleife
 8A25   58          CLI
 8A26   AD 8D 02    LDA  SHCOCTFL   ;Shift-Flag
 8A29   C9 02       CMP  #$02       ;Commodore-Taste ?
 8A2B   D0 0C       BNE  $8A39      ;nein, dann weiter
 8A2D   A2 00       LDX  #$00
 8A2F   A0 00       LDY  #$00
```

```
 8A31   E8          INX             ;innerer Wartezähler
 8A32   D0 FD       BNE   $8A31     ;innere Schleife
 8A34   C8          INY             ;äußerer Wartezähler
 8A35   C0 82       CPY   #$82      ;bis 130
 8A37   D0 F8       BNE   $8A31     ;äußere Schleife
 8A39   60          RTS

WEISSECK:
 8A3A   78          SEI
 8A3B   20 EC 82    JSR   BASROMAU
 8A3E   A9 D8       LDA   #$D8      ;Zeiger $20,$21 auf $D822
 8A40   85 21       STA   $21       ;(Farb-Ram oben rechts)
 8A42   A9 22       LDA   #$22
 8A44   85 20       STA   $20
 8A46   A0 00       LDY   #$00
 8A48   A2 00       LDX   #$00
 8A4A   A9 01       LDA   #$01      ;(weiß)
 8A4C   91 20       STA   ($20),Y   ;in 1 BS-Stelle speichern
 8A4E   C8          INY             ;nächstes Zeichen von 7
 8A4F   C0 06       CPY   #$06      ;7 Zeichen weiß
 8A51   D0 F7       BNE   $8A4A     ;'innere' Schleife
 8A53   20 CE 94    JSR   20PL40    ;$20,$21 um 40 erhöhen
 8A56   E8          INX             ;nächste Zeile
 8A57   A0 00       LDY   #$00
 8A59   E0 06       CPX   #$06      ;7 Zeilen voll ?
 8A5B   D0 ED       BNE   $8A4A     ;'äußere Schleife'
 8A5D   58          CLI
 8A5E   4C F3 82    JMP   BASROMEI

SIRQPLAY:                         ;IRQ-Unterroutine; wird auf-
                                  ;gerufen,wenn PLAYFLAG = $0A
 8A61   AD 8A CB    LDA   MUSICZ4
 8A64   C9 00       CMP   #$00
 8A66   F0 1C       BEQ   $8A84
 8A68   CE 86 CB    DEC   MUSICZ2
 8A6B   AD 86 CB    LDA   MUSICZ2
 8A6E   C9 00       CMP   #$00
 8A70   F0 01       BEQ   $8A73
 8A72   60          RTS

 8A73   CE 8A CB    DEC   MUSICZ4
 8A76   AD 8A CB    LDA   MUSICZ4
 8A79   C9 00       CMP   #$00
 8A7B   F0 07       BEQ   $8A84
 8A7D   AD 88 CB    LDA   MUSICZ3
 8A80   8D 86 CB    STA   MUSICZ2
 8A83   60          RTS

 8A84   AD 9B CB    LDA   MUSICADR   ;Adresse des MUSIC-String
 8A87   85 FC       STA   $FC
 8A89   AD 9C CB    LDA   MUSICADR+1
```

```
8A8C  85 FD       STA  $FD
8A8E  AC 9D CB    LDY  MUSICZ1   ;bis wieweit String bearb.
8A91  B1 FC       LDA  ($FC),Y   ;akt. Zeichen aus String
8A93  C9 85       CMP  #$85      ;(F1)
8A95  90 18       BCC  $8AAF     ;kleiner "F1",dann weiter
8A97  C9 8D       CMP  #$8D      ;$8C = (F8)
8A99  B0 14       BCS  $8AAF     ;größer "F8",dann weiter
8A9B  38          SEC
8A9C  B1 FC       LDA  ($FC),Y   ;Zeichenkode
8A9E  E9 84       SBC  #$84      ;minus $84
8AA0  8D 86 CB    STA  MUSICZ2   ;Nummer der Fkt.-Taste
8AA3  8D 88 CB    STA  MUSICZ3
8AA6  AD 96 CB    LDA  MUSICDAU
8AA9  8D 8A CB    STA  MUSICZ4
8AAC  4C D4 8A    JMP  $8AD4

8AAF  C9 93       CMP  #$93      ;(CLR)
8AB1  D0 03       BNE  $8AB6     ;kein (CLR), dann weiter
8AB3  4C F0 8A    JMP  $8AF0

8AB6  8C 9F CB    STY  MUSICZ5   ;aktuelle Stimme
8AB9  20 A1 8B    JSR  NOTESUCH  ;High/Low in A/Y
8ABC  AC 8C CB    LDY  MREGADR
8ABF  84 FE       STY  $FE
8AC1  AC 8D CB    LDY  MREGADR+1
8AC4  84 FF       STY  $FF
8AC6  A0 01       LDY  #$01
8AC8  91 FE       STA  ($FE),Y   ;High-Byte setzen
8ACA  8A          TXA
8ACB  88          DEY
8ACC  91 FE       STA  ($FE),Y   ;Low-Byte setzen
8ACE  20 1A 8C    JSR  STIMMEIN
8AD1  AC 9F CB    LDY  MUSICZ5
8AD4  CC 9E CB    CPY  MUSICLEN
8AD7  B0 05       BCS  PLAYEND   ;fertig ?
8AD9  C8          INY
8ADA  8C 9D CB    STY  MUSICZ1   ;Zaehler erhoeht
8ADD  60          RTS            ;fuer diesmal fertig

PLAYEND:
8ADE  A9 00       LDA  #$00
8AE0  8D 9D CB    STA  MUSICZ1
8AE3  8D 91 CB    STA  PLAYFLAG ;'Stueck fertig'
8AE6  60          RTS

8AE7  4C 6E 8B    JMP  $8B6E

8AEA  4C 7F 8B    JMP  $8B7F

8AED  4C 90 8B    JMP  $8B90
```

```
8AF0  C8        INY                ;naechstes Zeichen
8AF1  B1 FC     LDA  ($FC),Y       ;aus Music-String
8AF3  A2 00     LDX  #$00
8AF5  8E 8A CB  STX  MUSICZ4
8AF8  C9 31     CMP  #$31          ;"1" (Stimme 1)
8AFA  F0 EB     BEQ  $8AE7         ;ja, dann nach $8B6E
8AFC  C9 32     CMP  #$32          ;"2" (Stimme 2)
8AFE  F0 EA     BEQ  $8AEA         ;ja, dann nach $8B7F
8B00  C9 33     CMP  #$33          ;"3" (Stimme 3)
8B02  F0 E9     BEQ  $8AED         ;ja, dann nach $8B90
8B04  C9 47     CMP  #$47          ;"G" (Stimme aus)
8B06  F0 47     BEQ  $8B4F         ;ja, dann nach $8B4F
8B08  48        PHA
8B09  8C 9F CB  STY  MUSICZ5
8B0C  AC 8C CB  LDY  MREGADR
8B0F  84 FE     STY  $FE
8B11  AC 8D CB  LDY  MREGADR+1
8B14  84 FF     STY  $FF
8B16  AC 9F CB  LDY  MUSICZ5
8B19  68        PLA
8B1A  C9 54     CMP  #$54          ;"T" (Sync-Bit invert.)
8B1C  D0 13     BNE  $8B31         ;nein, dann weiter
8B1E  98        TYA
8B1F  48        PHA
8B20  AC A8 CB  LDY  STIMMENR     ;Stimme (0,1,2)
8B23  B9 4A C6  LDA  WAVETAB,Y    ;alte Wellenform
8B26  49 02     EOR  #$02          ;Bit 1 invertieren
8B28  A0 04     LDY  #$04          ;in Wellenform-
8B2A  91 FE     STA  ($FE),Y       ;Register schreiben
8B2C  68        PLA
8B2D  A8        TAY
8B2E  4C D4 8A  JMP  $8AD4         ;naechstes Zeichen

8B31  C9 43     CMP  #$43          ;"C" (Wellenform = 0)
8B33  D0 0A     BNE  $8B3F         ;nein, dann weiter
8B35  98        TYA
8B36  48        PHA
8B37  AC A8 CB  LDY  STIMMENR
8B3A  A9 00     LDA  #$00          ;Wellenform auf 0
8B3C  4C 28 8B  JMP  $8B28         ;s.o.

8B3F  C9 52     CMP  #$52          ;"R" (Repeat/Wiederholung)
8B41  D0 09     BNE  $8B4C         ;nein, naechstes Zeichen
8B43  A0 00     LDY  #$00
8B45  8C 9D CB  STY  MUSICZ1       ;Zaehler f r abgearbei-
8B48  8C 8A CB  STY  MUSICZ4       ;tete Zeichen
8B4B  60        RTS

8B4C  4C D4 8A  JMP  $8AD4         ;naechstes Zeichen bearb.
```

```
; Stimme aus ("G")
 8B4F   8C 9F CB   STY  MUSICZ5
 8B52   AC 8C CB   LDY  MREGADR
 8B55   84 FE      STY  $FE
 8B57   AC 8D CB   LDY  MREGADR+1
 8B5A   84 FF      STY  $FF
 8B5C   AC A8 CB   LDY  STIMMENR
 8B5F   B9 4A C6   LDA  WAVETAB,Y
 8B62   29 FE      AND  #$FE       ;Key-Bit loeschen
 8B64   A0 04      LDY  #$04
 8B66   91 FE      STA  ($FE),Y
 8B68   AC 9F CB   LDY  MUSICZ5
 8B6B   4C D4 8A   JMP  $8AD4      ;naechstes Zeichen

; auf Stimme 1 schalten
 8B6E   A9 D4      LDA  #$D4
 8B70   8D 8D CB   STA  MREGADR+1
 8B73   A9 00      LDA  #$00
 8B75   8D 8C CB   STA  MREGADR
 8B78   A9 00      LDA  #$00
 8B7A   8D A8 CB   STA  STIMMENR
 8B7D   F0 CD      BEQ  $8B4C      ;unbedingter Sprung

; auf Stimme 2 schalten
 8B7F   A9 D4      LDA  #$D4
 8B81   8D 8D CB   STA  MREGADR+1
 8B84   A9 07      LDA  #$07
 8B86   8D 8C CB   STA  MREGADR
 8B89   A9 01      LDA  #$01
 8B8B   8D A8 CB   STA  STIMMENR
 8B8E   D0 BC      BNE  $8B4C      ;unbedingter Sprung

; auf Stimme 3 schalten
 8B90   A9 D4      LDA  #$D4
 8B92   8D 8D CB   STA  MREGADR+1
 8B95   A9 0E      LDA  #$0E
 8B97   8D 8C CB   STA  MREGADR
 8B9A   A9 02      LDA  #$02
 8B9C   8D A8 CB   STA  STIMMENR
 8B9F   D0 AB      BNE  $8B4C      ;unbedingter Sprung

NOTESUCH:
 8BA1   A0 00      LDY  #$00
 8BA3   D9 DE 8B   CMP  NOTENTAB,Y ;in Tabelle ?
 8BA6   F0 09      BEQ  $8BB1      ;ja, dann zu $8BB1
 8BA8   C8         INY             ;naechstes Zeichen in Tab.
 8BA9   C0 14      CPY  #$14       ;Tabelle zu Ende ?
 8BAB   D0 F6      BNE  $8BA3      ;nein, dann neachst.
 8BAD   A9 00      LDA  #$00       ;sonst Akku und X-Reg.
 8BAF   AA         TAX             ;annullieren
 8BB0   60         RTS
```

```
8BB1  B9 06 8C   LDA  FRQTABH,Y ;Frequenzwerte f r
8BB4  8D 9A C5   STA  MFREQ+1   ;niedrigste Oktave aus
8BB7  B9 F2 8B   LDA  FRQTABL,Y ;Tabelle holen
8BBA  8D 99 C5   STA  MFREQ     ;und nach MFREQ,MFREQ+1
8BBD  AC 9F CB   LDY  MUSICZ5   ;Zeiger in Music-String
8BC0  C8         INY            ;erh hen
8BC1  8C 9F CB   STY  MUSICZ5
8BC4  B1 FC      LDA  ($FC),Y   ;Oktavzeichen (0-7)
8BC6  E9 30      SBC  #$30      ;$30 vom ASC-Code subtr.
8BC8  AA         TAX            ;gibt Wert von Ziffer
8BC9  E0 00      CPX  #$00      ;Null ?
8BCB  F0 0A      BEQ  $8BD7     ;dann fertig
8BCD  0E 99 C5   ASL  MFREQ     ;Frequenz
8BD0  2E 9A C5   ROL  MFREQ+1   ;verdoppeln
8BD3  CA         DEX            ;sooft, wie Oktavnummer
8BD4  4C C9 8B   JMP  $8BC9

8BD7  AD 9A C5   LDA  MFREQ+1  ;A/X mit Frequenz laden
8BDA  AE 99 C5   LDX  MFREQ
8BDD  60         RTS

NOTENTAB:
 8BDE  43 C3 44 C4 45 46 C6 47    cCdDefFg
 8BE6  C7 41 C1 42 5A             GaAbz
 8BEB  AC B1 BB A5 B0 BF BC       cdefgab  (mit C= Taste)

FRQTABL:
 8BF2  12 23 34 46 5A 6E 84 9B   ;Frequenzwerte f r Tasten
 8BFA  B3 CD E9 06 00 23 46 5A   ;Low-Bytes
 8C02  84 B3 E9 06

FRQTABH:
 8C06  01 01 01 01 01 01 01 01   ;Frequenzwerte f r Tasten
 8C0E  01 01 01 02 00 01 01 01   ;High-Bytes
 8C16  01 01 01 02

STIMMEIN:
 8C1A  AC A8 CB   LDY  STIMMENR
 8C1D  B9 A4 CB   LDA  ADTAB,Y    ;Attack/Decay-Wert
 8C20  A0 05      LDY  #$05
 8C22  91 FE      STA  ($FE),Y    ;setzen
 8C24  AC A8 CB   LDY  STIMMENR
 8C27  B9 A0 CB   LDA  SRTAB,Y    ;Sustain/Release-Wert
 8C2A  A0 06      LDY  #$06
 8C2C  91 FE      STA  ($FE),Y    ;setzen
 8C2E  AC A8 CB   LDY  STIMMENR
 8C31  B9 4A C6   LDA  WAVETAB,Y ;Wellenform
 8C34  29 FE      AND  #$FE       ;Key-Bit loeschen
 8C36  A0 04      LDY  #$04
 8C38  91 FE      STA  ($FE),Y
 8C3A  A0 04      LDY  #$04
```

```
 8C3C  09 01       ORA  #$01      ;Key-Bit setzen
 8C3E  91 FE       STA  ($FE),Y
 8C40  60          RTS

MOBEXS:                           ;setzt Vergrößerung X
 8C41  AE BC C5    LDX  MOBNR
 8C44  AD 1D D0    LDA  VICMEX
 8C47  1D 78 97    ORA  MBITTABS,X
 8C4A  8D 1D D0    STA  VICMEX
 8C4D  60          RTS

MOBEYS:                           ;setzt Vergrößerung Y
 8C4E  AE BC C5    LDX  MOBNR
 8C51  AD 17 D0    LDA  VICMEY
 8C54  1D 78 97    ORA  MBITTABS,X
 8C57  8D 17 D0    STA  VICMEY
 8C5A  60          RTS

MOBEXL:                           ;löscht Vergrößerung X
 8C5B  AE BC C5    LDX  MOBNR
 8C5E  AD 1D D0    LDA  VICMEX
 8C61  3D 80 97    AND  MBITTABL,X
 8C64  8D 1D D0    STA  VICMEX
 8C67  60          RTS

MOBEYL:                           ;löscht Vergrößerung Y
 8C68  AE BC C5    LDX  MOBNR
 8C6B  AD 17 D0    LDA  VICMEY
 8C6E  3D 80 97    AND  MBITTABL,X
 8C71  8D 17 D0    STA  VICMEY
 8C74  60          RTS

GETBYTC:                          ;prüft Komma und holt Byte
 8C75  20 FD AE    JSR  CHKCOM
 8C78  4C 9E B7    JMP  $B79E     ;GETBYT

SGETSTR1:                    ;'Verzierter' Aufruf von SGETSTR
 8C7B  AD 96 C5    LDA  PORTSPEI
 8C7E  48          PHA
 8C7F  20 B7 86    JSR  SGETSTR
 8C82  8D 7D C5    STA  ZWSPEI1
 8C85  68          PLA
 8C86  8D 96 C5    STA  PORTSPEI
 8C89  AD 7D C5    LDA  ZWSPEI1
 8C8C  60          RTS

SGETARI:                        ;holt arithmetisches Element
 8C8D  A9 00       LDA  #$00
 8C8F  85 0D       STA  $0D       ;Typflag auf numerisch
 8C91  20 73 00    JSR  CHRGET    ;Zeichen holen
 8C94  B0 03       BCS  $8C99     ;keine Ziffer, dann weiter
```

```
 8C96  4C 79 80   JMP  SASCFLP  ;Text nach FAC wandeln

 8C99  C9 24      CMP  #$24     ;"$"    (Hex-Zahl folgt)
 8C9B  F0 3A      BEQ  GETHEX   ;Hex-Zahl
 8C9D  C9 25      CMP  #$25     ;"%"  (Binärzahl folgt)
 8C9F  F0 22      BEQ  GETBIN   ;Binärzahl, dann zu $8CC3
 8CA1  C9 64      CMP  #$64     ;(Simon's Funktion folgt)
 8CA3  F0 03      BEQ  $8CA8    ;ja, dann zu 8DDC
 8CA5  4C 7C 80   JMP  SGETARI1 ;zur normalen Basic-
                                ;Routine GETARI

 8CA8  4C DC 8D   JMP  $8DDC    ;zur Ausw. v. Simon's Fkt.

CHHEXCHR:                       ;prüfe ob Zeichen in
                                ;"0123456789ABCDEF" enth.
 8CAB  C9 30      CMP  #$30     ;"0"
 8CAD  90 0C      BCC  $8CBB    ;kleiner "0", dann Fehler
 8CAF  C9 47      CMP  #$47     ;"G"
 8CB1  B0 08      BCS  $8CBB    ;größer "G", dann Fehler
 8CB3  C9 3A      CMP  #$3A     ;":"
 8CB5  90 0B      BCC  $8CC2    ;kleiner="9", dann weiter
 8CB7  C9 41      CMP  #$41     ;"A"
 8CB9  B0 07      BCS  $8CC2    ;größer= "A", dann weiter
 8CBB  68         PLA           ;Rücksprungadresse vom
 8CBC  68         PLA           ;Stack entfernen
 8CBD  A9 05      LDA  #$05     ;Nr. für "not hex char"
 8CBF  4C 8C 88   JMP  SERROUT  ;Fehlermeldung ausgeben

 8CC2  60         RTS

GETBIN:                         ;holt Binärwert
 8CC3  20 73 00   JSR  CHRGET   ;überliest Code
 8CC6  20 09 9A   JSR  BINCON   ;holt Binärzahl nach $A8
 8CC9  A4 A8      LDY  $A8      ;Low-Byte
 8CCB  A2 00      LDX  #$00     ;High-Byte

SXYFLP:
 8CCD  84 63      STY  $63      ;Low-Byte
 8CCF  86 62      STX  $62      ;High-Byte
 8CD1  A2 90      LDX  #$90     ;Exponent
 8CD3  38         SEC
 8CD4  4C 49 BC   JMP  XFLP     ;wandelt nach Fließkomma

GETHEX:                         ;holt Hexadezimalwert
 8CD7  20 73 00   JSR  CHRGET   ;überliest Code
 8CDA  20 E0 8C   JSR  HEXCON   ;holt Hex-Zahl
 8CDD  4C 1E 8D   JMP  $8D1E    ;Fortsetzung bei $8D1E

HEXCON:
 8CE0  A0 00      LDY  #$00
 8CE2  B1 7A      LDA  ($7A),Y  ;akt. Basic-Zeichen
```

```
 8CE4   20 AB 8C    JSR   CHHEXCHR
 8CE7   8D 1F C5    STA   POS        ;zwischenspeichern
 8CEA   20 73 00    JSR   CHRGET     ;nächstes Zeichen lesen
 8CED   B1 7A       LDA   ($7A),Y    ;akt. Zeichen holen
 8CEF   20 AB 8C    JSR   CHHEXCHR
 8CF2   A8          TAY
 8CF3   AD 1F C5    LDA   POS
 8CF6   20 28 8D    JSR   HEX2W      ;Wert von 2-st Hexz. n. A
 8CF9   85 A9       STA   $A9        ;in $A9 speichern
 8CFB   20 73 00    JSR   CHRGET     ;ab hier anaolg für die
 8CFE   A0 00       LDY   #$00       ;hinteren beiden Zeichen
 8D00   B1 7A       LDA   ($7A),Y
 8D02   20 AB 8C    JSR   CHHEXCHR
 8D05   8D 1F C5    STA   POS
 8D08   20 73 00    JSR   CHRGET
 8D0B   B1 7A       LDA   ($7A),Y
 8D0D   20 AB 8C    JSR   CHHEXCHR
 8D10   A8          TAY
 8D11   AD 1F C5    LDA   POS
 8D14   20 28 8D    JSR   HEX2W
 8D17   85 63       STA   $63        ;Low-Byte
 8D19   A5 A9       LDA   $A9        ;High-Byte
 8D1B   85 62       STA   $62
 8D1D   60          RTS

;Fortsetzung von GETHEX
 8D1E   20 73 00    JSR   CHRGET
 8D21   A2 90       LDX   #$90
 8D23   38          SEC
 8D24   20 49 BC    JSR   XFLP       ;wandelt nach Fließkomma
 8D27   60          RTS

HEX2W:                   ;holt Wert v. 2-stell. Hexzahl in Akku
 8D28   48          PHA              ;1.Ziffer merken
 8D29   98          TYA              ;2.Ziffer
 8D2A   20 3C 8D    JSR   HEXW       ;umwandeln
 8D2D   8D 54 CB    STA   ZWISP2     ;Ergebnis merken
 8D30   68          PLA              ;1.Ziffer
 8D31   20 3C 8D    JSR   HEXW       ;umwandeln
 8D34   0A          ASL   A          ; mal 16
 8D35   0A          ASL   A
 8D36   0A          ASL   A
 8D37   0A          ASL   A
 8D38   0D 54 CB    ORA   ZWISP2     ;plus Wert v. 2.Ziffer
 8D3B   60          RTS

HEXW:
 8D3C   38          SEC
 8D3D   E9 30       SBC   #$30       ;Code minus $30
 8D3F   C9 0A       CMP   #$0A
 8D41   90 02       BCC   $8D45      ;größer als 10,
```

```
 8D43  E9 07     SBC  #$07     ;dann noch 7 abziehen
 8D45  60        RTS

FNEXOR:
 8D46  20 73 00  JSR  CHRGET   ;Code ueberlesen
 8D49  20 FA AE  JSR  CHKKLA   ;Klammer auf ?
 8D4C  20 15 94  JSR  SGETADR2 ;holt Adresswert
 8D4F  84 AA     STY  $AA      ;zwischenspeichern
 8D51  85 AB     STA  $AB      ;in $AA,$AB
 8D53  20 FD AE  JSR  CHKCOM   ;Komma ?
 8D56  20 15 94  JSR  SGETADR2 ;Adresswert holen
 8D59  20 F7 AE  JSR  CHKKLZ   ;Klammer zu ?
 8D5C  A5 65     LDA  $65      ;Low-Byte
 8D5E  45 AA     EOR  $AA      ;verknuepfen
 8D60  A8        TAY
 8D61  A5 64     LDA  $64      ;High-Byte
 8D63  45 AB     EOR  $AB
 8D65  AA        TAX
 8D66  4C CD 8C  JMP  SXYFLP   ;zurueck nach Fliesskomma

FNDUP:
 8D69  20 73 00  JSR  CHRGET   ;Code ueberlesen
 8D6C  20 FA AE  JSR  CHKKLA   ;Klammer auf ?
 8D6F  20 7B 8C  JSR  SGETSTR1 ;String holen
 8D72  A4 69     LDY  $69      ;Stringlaenge nach POS
 8D74  8C 1F C5  STY  POS
 8D77  8D AC CB  STA  STR1+1 ;Stringadresse speichern
 8D7A  8E AB CB  STX  STR1
 8D7D  20 FD AE  JSR  CHKKOM   ;Komma ?
 8D80  20 15 94  JSR  SGETADR2 ;Wert holen
 8D83  20 F7 AE  JSR  CHKKLZ   ;Klammer zu ?
 8D86  A6 65     LDX  $65      ;Low-Byte
 8D88  8E AE CB  STX  DWERT    ;als Wert speichern
 8D8B  A0 00     LDY  #$00
 8D8D  8C 20 C5  STY  POS1     ;POS1 fuer Gesamtlaenge
 8D90  A0 00     LDY  #$00
 8D92  EE 20 C5  INC  POS1
 8D95  C8        INY
 8D96  CC 1F C5  CPY  POS
 8D99  D0 F7     BNE  $8D92    ;innere Schleife
 8D9B  CA        DEX           ;Wiederholungszaehler
 8D9C  D0 F2     BNE  $8D90    ;aeussere Schleife
 8D9E  AD 20 C5  LDA  POS1     ;neue Gesamtlaenge
 8DA1  20 7D B4  JSR  NEUSTR   ;String einrichten
 8DA4  AD AC CB  LDA  STR1+1
 8DA7  85 21     STA  $21
 8DA9  AD AB CB  LDA  STR1
 8DAC  85 20     STA  $20
 8DAE  AE AE CB  LDX  DWERT    ;X als Wiederh.-Zaehler
 8DB1  A0 00     LDY  #$00
 8DB3  8C 18 C5  STY  STRZ1    ;Zaehler im neuen String
```

```
8DB6  B1 20     LDA  ($20),Y   ;Zeichen aus altem String
8DB8  8C 19 C5  STY  STRZ2     ;Zaehler im alten String
8DBB  AC 18 C5  LDY  STRZ1
8DBE  91 62     STA  ($62),Y   ;Zeichen des neuen Strings
8DC0  C8        INY            ;Zaehler fuer neuen String
8DC1  8C 18 C5  STY  STRZ1     ;erhoehen
8DC4  AC 19 C5  LDY  STRZ2     ;alter String
8DC7  C8        INY
8DC8  CC 1F C5  CPY  POS       ;Ende erreicht ?
8DCB  D0 E9     BNE  $8DB6     ;nein,dann naechst. Zeich.
8DCD  A0 00     LDY  #$00      ;Zaehler im alten String
8DCF  8C 20 C5  STY  POS1      ;(unsinnig)
8DD2  CA        DEX            ;Wiederholungszaehler
8DD3  D0 E1     BNE  $8DB6     ;aeussere Schleife
8DD5  A9 FF     LDA  #$FF
8DD7  85 0D     STA  $0D       ;Stringflag setzen
8DD9  4C CA B4  JMP  PUSHSTR   ;String auf Stapel legen

;arithmetischen Ausdruck hinter $64-Code auswerten
8DDC  20 73 00  JSR  CHRGET    ;Code ueberlesen
8DDF  A0 00     LDY  #$00
8DE1  B1 7A     LDA  ($7A),Y   ;naechsten Code
8DE3  C9 55     CMP  #$55      ;Code fuer DUP
8DE5  D0 03     BNE  $8DEA     ;nein, dann weiter
8DE7  4C 69 8D  JMP  FNDUP

8DEA  C9 5A     CMP  #$5A      ;Code fuer EXOR
8DEC  D0 03     BNE  $8DF1     ;nein, dann weiter
8DEE  4C 46 8D  JMP  FNEXOR

8DF1  C9 51     CMP  #$51      ;Code fuer JOY
8DF3  D0 03     BNE  $8DF8     ;nein, dann weiter
8DF5  4C D4 88  JMP  FNJOY

8DF8  C9 0E     CMP  #$0E      ;Code fuer FRAC
8DFA  D0 4C     BNE  $8E48     ;nein, dann weiter

FNFRAC:
8DFC  20 73 00  JSR  CHRGET    ;Code ueberlesen
8DFF  A5 7A     LDA  $7A       ;Programmzaehler sichern
8E01  48        PHA
8E02  A5 7B     LDA  $7B
8E04  48        PHA
8E05  A9 00     LDA  #$00
8E07  85 62     STA  $62       ;null
8E09  85 63     STA  $63
8E0B  A2 90     LDX  #$90
8E0D  38        SEC
8E0E  20 49 BC  JSR  XFLP      ;nach Fliesskomma wandeln
8E11  20 0C BC  JSR  FACARG    ;FAC nach ARG uebertragen
8E14  20 FA AE  JSR  CHKKLA    ;Klammer auf ?
```

```
8E17  20 9E AD   JSR  FRMEVL    ;holt Ausdruck
8E1A  A5 66      LDA  $66       ;Vorzeichen von FAC
8E1C  85 AA      STA  $AA       ;in $AA speichern
8E1E  30 03      BMI  $8E23     ;negativ ?
8E20  4C 26 8E   JMP  $8E26
8E23  20 B4 BF   JSR  $BFB4     ;Vorzeichen wechseln
8E26  20 F7 AE   JSR  CHKKLZ    ;Klammer zu ?
8E29  20 1B BC   JSR  FACRUND   ;FAC runden
8E2C  20 0C BC   JSR  FACARG    ;FAC nach ARG uebertragen
8E2F  20 CC BC   JSR  $BCCC     ;Basic-Funktion INT
8E32  68         PLA            ;alten Programmzaehler
8E33  68         PLA            ;vom Stapel entfernen
8E34  20 53 B8   JSR  FAMINUS   ;FAC = ARG - FAC
8E37  A9 00      LDA  #$00
8E39  85 0D      STA  $0D       ;Stringflag loeschen
8E3B  A5 AA      LDA  $AA       ;altes Vorzeichen
8E3D  30 03      BMI  $8E42     ;negativ ?
8E3F  4C 45 8E   JMP  $8E45
8E42  20 B4 BF   JSR  $BFB4     ;Vorzeichen wechseln
8E45  4C 1B BC   JMP  FACRUND   ;FAC runden

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

8E48  C9 10      CMP  #$10      ;Code fuer PLACE
8E4A  F0 03      BEQ  FNPLACE   ;ja, dann zu FNPLACE
8E4C  4C AC 8E   JMP  $8EAC     ;sonst weiter

FNPLACE:
8E4F  20 73 00   JSR  CHRGET    ;Code ueberlesen
8E52  20 FA AE   JSR  CHKKLA    ;(
8E55  20 7B 8C   JSR  SGETSTR1  ;String holen
8E58  85 A8      STA  $A8       ;Stringadresse
8E5A  86 A7      STX  $A7       ;nach $A7,$A8
8E5C  A5 69      LDA  $69       ;Stringlaenge
8E5E  8D B7 CB   STA  STRLEN1   ;speichern
8E61  20 FD AE   JSR  CHKCOM    ;Komma ?
8E64  20 7B 8C   JSR  SGETSTR1
8E67  85 AA      STA  $AA       ;Stringadresse
8E69  86 A9      STX  $A9       ;nach $A9,$AA
8E6B  A5 69      LDA  $69       ;Stringlaenge
8E6D  8D B4 CB   STA  STRLEN2   ;speichern
8E70  20 F7 AE   JSR  CHKKLZ    ;)
8E73  A0 00      LDY  #$00
8E75  8C B1 CB   STY  PLPOS     ;Ergebnis
8E78  B1 A7      LDA  ($A7),Y   ;Zeichen aus 1. String
8E7A  D1 A9      CMP  ($A9),Y   ;= Zeichen aus 2. String ?
8E7C  D0 16      BNE  $8E94     ;wenn nein, dann weiter
8E7E  C8         INY            ;naechstes Zeichen
8E7F  CC B7 CB   CPY  STRLEN1   ;Stringende erreicht ?
8E82  D0 F4      BNE  $8E78     ;nein,dann naechst. Zeich.
8E84  EE B1 CB   INC  PLPOS     ;Position erhoehen
```

```
 8E87  AD B1 CB   LDA  PLPOS     ;Ergebnis
 8E8A  A8         TAY
 8E8B  A9 00      LDA  #$00
 8E8D  85 0D      STA  $0D       ;Stringflag loeschen
 8E8F  A2 00      LDX  #$00
 8E91  4C CD 8C   JMP  SXYFLP    ;nach Fliesskomma

 8E94  A0 00      LDY  #$00      ;Zaehler ruecksetzen
 8E96  E6 A9      INC  $A9       ;$A9,$AA erhoehen
 8E98  D0 02      BNE  $8E9C
 8E9A  E6 AA      INC  $AA
 8E9C  EE B1 CB   INC  PLPOS     ;Position erhoehen
 8E9F  AD B1 CB   LDA  PLPOS
 8EA2  CD B4 CB   CMP  STRLEN2   ;Stringlaenge erreicht ?
 8EA5  D0 D1      BNE  $8E78     ;Schleife
 8EA7  A9 00      LDA  #$00      ;String nicht gefunden
 8EA9  4C 8A 8E   JMP  $8E8A     ;s.o.

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 8EAC  C9 5B      CMP  #$5B      ;Code fuer INSERT
 8EAE  F0 03      BEQ  FNINSERT  ;wenn ja, dann dahin
 8EB0  4C 8A 8F   JMP  $8F8A     ;sonst weiter

FNINSERT:
 8EB3  20 73 00   JSR  CHRGET    ;Code ueberlesen
 8EB6  20 FA AE   JSR  CHKKLA    ;(
 8EB9  20 7B 8C   JSR  SGETSTR1  ;holt einzusetzenden Str.
 8EBC  85 A8      STA  $A8       ;Stringadresse
 8EBE  86 A7      STX  $A7       ;nach $A7,$A8
 8EC0  A5 69      LDA  $69       ;Stringlaenge
 8EC2  8D B7 CB   STA  STRLEN1
 8EC5  20 FD AE   JSR  CHKCOM    ;Komma ?
 8EC8  20 7B 8C   JSR  SGETSTR1  ;holt zweiten String
 8ECB  85 AA      STA  $AA       ;Stringadresse
 8ECD  86 A9      STX  $A9       ;nach $A9,$AA
 8ECF  A5 69      LDA  $69       ;Stringlaenge
 8ED1  8D B4 CB   STA  STRLEN2
 8ED4  20 75 8C   JSR  GETBYTC   ;holt Byte-Wert
 8ED7  8E 1F C5   STX  POS       ;in POS speichern
 8EDA  20 F7 AE   JSR  CHKKLZ    ;)
 8EDD  18         CLC
 8EDE  AD B4 CB   LDA  STRLEN2   ;Laenge 2
 8EE1  6D B7 CB   ADC  STRLEN1   ;+Laenge 1
 8EE4  90 0A      BCC  $8EF0     ;kleiner 256, dann weiter
 8EE6  A9 03      LDA  #$03      ;Nr. f. 'string to large'
 8EE8  4C 8C 88   JMP  SERROUT   ;Fehlermeldung ausgeben

 8EEB  A9 02      LDA  #$02      ;Nr. f. 'insert too large'
 8EED  4C 8C 88   JMP  SERROUT   ;Fehlermeldung ausgeben
```

```
8EF0  20 7D B4  JSR  NEUSTR     ;neuen String einrichten
8EF3  A0 00     LDY  #$00
8EF5  CC 1F C5  CPY  POS        ;Position erreicht ?
8EF8  F0 0C     BEQ  $8F06      ;ja, dann hier fertig
8EFA  B1 A9     LDA  ($A9),Y    ;Zeichen aus altem String
8EFC  91 62     STA  ($62),Y    ;in neuen String eintragen
8EFE  C8        INY             ;naechstes Zeichen
8EFF  CC B4 CB  CPY  STRLEN2    ;String zu Ende ?
8F02  F0 E7     BEQ  $8EEB      ;ja, dann Fehler
8F04  D0 EF     BNE  $8EF5      ;sonst weiter

8F06  8C 20 C5  STY  POS1       ;Position nach POS1
8F09  8C AE CB  STY  DWERT      ;und DWERT
8F0C  A0 00     LDY  #$00
8F0E  8C AB CB  STY  STR1      ;STR1 als Zaehler im 1.Str.
8F11  B1 A7     LDA  ($A7),Y    ;Zeichen aus 1.String
8F13  EE AB CB  INC  STR1       ;Zaehler erhoehen
8F16  AC 20 C5  LDY  POS1       ;Position holen
8F19  91 62     STA  ($62),Y    ;Zeichen speichern
8F1B  EE 20 C5  INC  POS1       ;Position erhoehen
8F1E  AC AB CB  LDY  STR1       ;Ende von 1.String
8F21  CC B7 CB  CPY  STRLEN1    ;erreicht ?
8F24  D0 EB     BNE  $8F11      ;nein, dann Schleife
8F26  AC AE CB  LDY  DWERT      ;alte Position
8F29  B1 A9     LDA  ($A9),Y    ;Zeichen aus 2.String
8F2B  EE AE CB  INC  DWERT      ;Position erhoehen
8F2E  AC 20 C5  LDY  POS1       ;neue Position holen
8F31  91 62     STA  ($62),Y    ;Zeichen speichern
8F33  EE 20 C5  INC  POS1       ;Position erhoehen
8F36  AC AE CB  LDY  DWERT      ;Ende von 2.String
8F39  CC B4 CB  CPY  STRLEN2    ;erreicht ?
8F3C  D0 EB     BNE  $8F29      ;nein, dann Schleife
8F3E  A9 FF     LDA  #$FF       ;Stringflag
8F40  85 0D     STA  $0D        ;setzen
8F42  4C CA B4  JMP  PUSHSTR    ;String auf Stapel legen

IDIV:                           ;Ganzzahlige Division
8F45  A9 00     LDA  #$00
8F47  8D C2 CB  STA  IDIVREST ;Rest annullieren
8F4A  8D C3 CB  STA  IDIVREST+1
8F4D  AD BB CB  LDA  IDIVSOR  ;Divisor = 0 ?
8F50  0D BC CB  ORA  IDIVSOR+1
8F53  D0 07     BNE  $8F5C      ;nein, dann weiter
8F55  68        PLA             ;Stack bereinigen
8F56  68        PLA
8F57  A2 14     LDX  #$14       ;Nr. f. 'division by zero'
8F59  6C 00 03  JMP  ($0300)    ;Fehlermeldung ausgeben

8F5C  A2 10     LDX  #$10         ;Bitzaehler auf 16
8F5E  2E C0 CB  ROL  IDIVDEND     ;Carry-Flag (von SBC bei
8F61  2E C1 CB  ROL  IDIVDEND+1 ;$8F57) in Ergebnis und
```

```
 8F64   2E C2 CB    ROL   IDIVREST   ;naechstes Bit des Divi-
 8F67   2E C3 CB    ROL   IDIVREST+1 ;denden in Rest schieben
 8F6A   38          SEC
 8F6B   AD C2 CB    LDA   IDIVREST ;Divisor von Rest abziehen
 8F6E   ED BB CB    SBC   IDIVSOR
 8F71   A8          TAY
 8F72   AD C3 CB    LDA   IDIVREST+1
 8F75   ED BC CB    SBC   IDIVSOR+1
 8F78   90 06       BCC   $8F80     ;wenn zu gross, dann nicht
 8F7A   8C C2 CB    STY   IDIVREST ;speichern
 8F7D   8D C3 CB    STA   IDIVREST+1
 8F80   CA          DEX             ;Bitzaehler vermindern
 8F81   D0 DB       BNE   $8F5E     ;Schleife
 8F83   2E C0 CB    ROL   IDIVDEND   ;Carry in Ergebnis
 8F86   2E C1 CB    ROL   IDIVDEND+1 ;rotieren
 8F89   60          RTS

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 8F8A   C9 53       CMP   #$53      ;Code fuer DIV
 8F8C   D0 2A       BNE   $8FB8     ;nein, dann weiter

FNDIV:
 8F8E   20 73 00    JSR   CHRGET      ;Code ueberlesen
 8F91   20 FA AE    JSR   CHKKLA      ;(
 8F94   20 15 94    JSR   SGETADR2    ;holt Adresswert
 8F97   8D C1 CB    STA   IDIVDEND+1 ;als Dividend speichern
 8F9A   8C C0 CB    STY   IDIVDEND
 8F9D   20 FD AE    JSR   CHKCOM      ;Komma ?
 8FA0   20 15 94    JSR   SGETADR2    ;holt Wert
 8FA3   8D BC CB    STA   IDIVSOR+1   ;als Divisor speichern
 8FA6   8C BB CB    STY   IDIVSOR
 8FA9   20 F7 AE    JSR   CHKKLZ      ;)
 8FAC   20 45 8F    JSR   IDIV        ;Division ausfuehren
 8FAF   AC C0 CB    LDY   IDIVDEND    ;Ergebnis
 8FB2   AE C1 CB    LDX   IDIVDEND+1
 8FB5   4C CD 8C    JMP   SXYFLP     ;nach Fliesskomma wandeln

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 8FB8   C9 52       CMP   #$52      ;Code fuer MOD
 8FBA   D0 2A       BNE   $8FE6     ;nein, dann weiter

FNMOD:
 8FBC   20 73 00    JSR   CHRGET      ;Code ueberlesen
 8FBF   20 FA AE    JSR   CHKKLA      ;(
 8FC2   20 15 94    JSR   SGETADR2    ;Wert holen
 8FC5   8D C1 CB    STA   IDIVDEND+1 ;als Dividend speichern
 8FC8   8C C0 CB    STY   IDIVDEND
 8FCB   20 FD AE    JSR   CHKCOM      ;Komma ?
 8FCE   20 15 94    JSR   SGETADR2    ;Wert holen
```

```
 8FD1   8D BC CB    STA   IDIVSOR+1   ;als Divisor speichern
 8FD4   8C BB CB    STY   IDIVSOR
 8FD7   20 F7 AE    JSR   CHKKLZ      ;)
 8FDA   20 45 8F    JSR   IDIV        ;Division durchfuehren
 8FDD   AC C2 CB    LDY   IDIVREST    ;Rest als Ergebnis
 8FE0   AE C3 CB    LDX   IDIVREST+1
 8FE3   4C CD 8C    JMP   SXYFLP     ;nach Fliesskomma wandeln

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 8FE6   C9 59       CMP   #$59      ;Code fuer LIN
 8FE8   D0 0E       BNE   $8FF8     ;nein, dann weiter

FNLIN:
 8FEA   20 DB 83    JSR   INCBASBZ
 8FED   38          SEC
 8FEE   20 F0 FF    JSR   $FFF0     ;Cursor-Position holen
 8FF1   8A          TXA             ;Cursor-Zeile in X
 8FF2   A8          TAY
 8FF3   A2 00       LDX   #$00      ;High-Byte = 0
 8FF5   4C CD 8C    JMP   SXYFLP    ;nach Fliesskomma wandeln

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 8FF8   C9 58       CMP   #$58      ;Code fuer TEST
 8FFA   D0 30       BNE   $902C     ;nein, dann weiter

FNTEST:
 8FFC   20 73 00    JSR   CHRGET    ;Code ueberlesen
 8FFF   20 FA AE    JSR   CHKKLA    ;(
 9002   20 15 94    JSR   SGETADR2 ;Wert (X-Koord.)
 9005   84 09       STY   $09       ;nach $09,$0A
 9007   85 0A       STA   $0A
 9009   20 FD AE    JSR   CHKCOM    ;Komma ?
 900C   20 15 94    JSR   SGETADR2 ;Wert (Y-Koord.)
 900F   84 A4       STY   $A4       ;nach $A4
 9011   20 F7 AE    JSR   CHKKLZ    ;)
 9014   A9 0B       LDA   #$0B      ;11
 9016   85 F7       STA   $F7       ;als Punktfarbe
 9018   20 EC 82    JSR   BASROMAU ;Basic aus
 901B   20 97 92    JSR   PUNKT     ;Punkt-Routine
 901E   A2 00       LDX   #$00
 9020   A4 90       LDY   $90       ;Test-Ergebnis
 9022   20 F3 82    JSR   BASROMEI  ;Basic ein
 9025   20 0E 94    JSR   KERROMEIN ;Kernal ein
 9028   58          CLI
 9029   4C CD 8C    JMP   SXYFLP    ;nach Fliesskomma wandeln
```

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

```
 902C  C9 57       CMP  #$57      ;Code für INST
 902E  F0 03       BEQ  FNINST    ;ja ?
 9030  4C A4 90    JMP  $90A4     ;sonst weiter

FNINST:
 9033  20 73 00    JSR  CHRGET    ;Code überlesen
 9036  20 FA AE    JSR  CHKKLA    ;(
 9039  20 7B 8C    JSR  SGETSTR1  ;1. String holen
 903C  85 A8       STA  $A8       ;Adresse nach $A7,$A8
 903E  86 A7       STX  $A7
 9040  A5 69       LDA  $69       ;Stringlänge
 9042  8D B7 CB    STA  STRLEN1
 9045  20 FD AE    JSR  CHKCOM    ;Komma ?
 9048  20 7B 8C    JSR  SGETSTR1  ;2. String holen
 904B  85 AA       STA  $AA       ;Adresse nach $A9,$AA
 904D  86 A9       STX  $A9
 904F  A5 69       LDA  $69       ;Stringlänge
 9051  8D B4 CB    STA  STRLEN2
 9054  20 75 8C    JSR  GETBYTC   ;holt Position
 9057  8E 1F C5    STX  POS
 905A  20 F7 AE    JSR  CHKKLZ    ;)
 905D  18          CLC
 905E  AD B7 CB    LDA  STRLEN1   ;STRLEN1 + POS
 9061  6D 1F C5    ADC  POS
 9064  CD B4 CB    CMP  STRLEN2   ;kleiner gleich STRLEN2 ?
 9067  90 08       BCC  $9071     ;ja, dann Länge = STRLEN2
 9069  4C 74 90    JMP  $9074     ;sonst Länge = Akku

 906C  A9 03       LDA  #$03      ;Nr. f. 'string to large'
 906E  4C 8C 88    JMP  SERROUT   ;Fehler ausgeben

 9071  AD B4 CB    LDA  STRLEN2   ;Länge = STRLEN2
 9074  20 7D B4    JSR  NEUSTR    ;String einrichten
 9077  A0 00       LDY  #$00      ;Zähler
 9079  B1 A9       LDA  ($A9),Y   ;Zeichen aus altem String
 907B  91 62       STA  ($62),Y   ;in neuen String übertr.
 907D  C8          INY            ;naechstes Zeichen
 907E  CC B4 CB    CPY  STRLEN2   ;String zu Ende ?
 9081  D0 F6       BNE  $9079     ;nein, dann Schleife
 9083  A0 00       LDY  #$00
 9085  8C 20 C5    STY  POS1      ;POS1 als Zähler
 9088  B1 A7       LDA  ($A7),Y   ;Zeichen aus 1.String
 908A  EE 20 C5    INC  POS1      ;Zähler erhoehen
 908D  AC 1F C5    LDY  POS       ;neue Pos. holen
 9090  91 62       STA  ($62),Y   ;Zeichen speichern
 9092  EE 1F C5    INC  POS       ;neue Pos. erhoehen
 9095  AC 20 C5    LDY  POS1
 9098  CC B7 CB    CPY  STRLEN1   ;1. String zu Ende ?
 909B  D0 EB       BNE  $9088     ;nein, dann Schleife
```

```
 909D   A9 FF       LDA   #$FF       ;Stringflag
 909F   85 0D       STA   $0D        ;setzen
 90A1   4C CA B4    JMP   PUSHSTR    ;String auf Stapel legen

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 90A4   C9 56       CMP   #$56       ;Code für INKEY
 90A6   D0 0B       BNE   $90B3      ;nein ?

FNINKEY:
 90A8   20 DB 83    JSR   INCBASBZ   ;Code überlesen
 90AB   AC D6 C5    LDY   INKEY      ;gespeicherter Wert
 90AE   A2 00       LDX   #$00       ;High-Byte = 0
 90B0   4C CD 8C    JMP   SXYFLP     ;nach Fliessk. wandeln

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 90B3   C9 5C       CMP   #$5C       ;Code für POT
 90B5   D0 03       BNE   $90BA      ;nein ?
 90B7   4C 1A 89    JMP   FNPOT      ;zur POT-Funktion
 90BA   C9 5D       CMP   #$5D       ;Code für PENX
 90BC   D0 03       BNE   $90C1      ;nein ?
 90BE   4C 3A 89    JMP   FNPENX     ;zur PENX-Funktion
 90C1   C9 5F       CMP   #$5F       ;Code für PENY
 90C3   D0 03       BNE   $90C8      ;nein ?
 90C5   4C 43 89    JMP   FNPENY     ;zur PENY-Funktion
 90C8   C9 61       CMP   #$61       ;Code für GRAPHICS
 90CA   D0 0A       BNE   $90D6      ;nein ?

FNGRAPHICS:
 90CC   20 DB 83    JSR   INCBASBZ   ;Code überlesen
 90CF   A2 D0       LDX   #$D0       ;Konstante $D000
 90D1   A0 00       LDY   #$00
 90D3   4C CD 8C    JMP   SXYFLP     ;nach Fliesskomma

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 90D6   C9 60       CMP   #$60       ;Code für SOUND
 90D8   D0 0A       BNE   $90E4      ;nein ?

FNSOUND:
 90DA   20 DB 83    JSR   INCBASBZ   ;Code überlesen
 90DD   A2 D4       LDX   #$D4       ;Konstante $D400
 90DF   A0 00       LDY   #$00
 90E1   4C CD 8C    JMP   SXYFLP     ;nach Fliessk.

;Fortsetzung 'arithm. Ausdruck hinter $64-Code'

 90E4   C9 28       CMP   #$28       ;Code für AT
 90E6   D0 03       BNE   $90EB      ;nein ?
 90E8   4C B7 97    JMP   FNAT       ;zum AT-Befehl
```

```
 90EB   C9 7C        CMP   #$7C        ;Code für CHECK
 90ED   D0 57        BNE   FNERR       ;nein, dann zu ERR-Fkt.

FNCHECK:
 90EF   20 DB 83     JSR   INCBASBZ   ;Code überlesen
 90F2   20 FA AE     JSR   CHKKLA     ;(
 90F5   20 15 94     JSR   SGETADR2   ;Wert (Sprite-Nr.) holen
 90F8   8C 1F C5     STY   POS        ;in POS speichern
 90FB   AD 15 C5     LDA   DETECTART  ;0 bzw. 1
 90FE   D0 33        BNE   $9133      ;wenn 1, dann weiter

;Kollision zwischen zwei Sprites behandeln
 9100   20 FD AE     JSR   CHKCOM     ;Komma ?
 9103   20 15 94     JSR   SGETADR2   ;2. Wert holen
 9106   8C 20 C5     STY   POS1       ;in POS1 speichern
 9109   20 F7 AE     JSR   CHKKLZ     ;)
 910C   AC 1F C5     LDY   POS        ;Nr. von 1. Sprite
 910F   AD 14 C5     LDA   DETECTERG   ;Ergebnis von DETECT
 9112   19 78 97     ORA   MBITTABS,Y ;mit entspr. Bit odern
 9115   CD 14 C5     CMP   DETECTERG   ;war Bit gesetzt ?
 9118   D0 15        BNE   $912F       ;nein ?
 911A   AC 20 C5     LDY   POS1        ;Nr. von 2. Sprite
 911D   AD 14 C5     LDA   DETECTERG
 9120   19 78 97     ORA   MBITTABS,Y ;entspr. Bit
 9123   CD 14 C5     CMP   DETECTERG   ;war auch dieses Bit 1 ?
 9126   D0 07        BNE   $912F       ;nein ?
 9128   A0 00        LDY   #$00        ;0
 912A   A2 00        LDX   #$00
 912C   4C CD 8C     JMP   SXYFLP      ;nach Fliessk. wandeln

 912F   A0 01        LDY   #$01       ;1
 9131   D0 F7        BNE   $912A      ;unbed. Sprung

;Kollision von 1 Sprite mit Hintergrund-Zeichen behandeln
 9133   20 F7 AE     JSR   CHKKLZ      ;)
 9136   AC 1F C5     LDY   POS         ;Sprite-Nr.
 9139   AD 14 C5     LDA   DETECTERG   ;Ergebnis von DETECT
 913C   19 78 97     ORA   MBITTABS,Y ;mit entspr. Bit odern
 913F   CD 14 C5     CMP   DETECTERG   ;war Bit gesetzt ?
 9142   D0 EB        BNE   $912F       ;nein, dann CHECK=1
 9144   F0 E2        BEQ   $9128       ;sonst CHECK=0

FNERR:                                ;saemtliche ERR-Befehle
 9146   20 DB 83     JSR   INCBASBZ   ;Code überlesen
 9149   A0 00        LDY   #$00
 914B   B1 7A        LDA   ($7A),Y    ;naechster Basic-Code
 914D   C9 4E        CMP   #$4E       ;"N" (für ERRN)
 914F   F0 1C        BEQ   FNERRN     ;ja ?
 9151   C9 4C        CMP   #$4C       ;"L" (für ERRLN)
 9153   F0 03        BEQ   FNERRLN    ;ja ?
 9155   4C 2A 82     JMP   ENDSMB     ;sonst Ende (und Fehler)
```

```
FNERRLN:
 9158   20 DB 83    JSR   INCBASBZ ;Code ("L") überlesen
 915B   B1 7A       LDA   ($7A),Y  ;naechster Basic-Code
 915D   C9 4E       CMP   #$4E     ;"N"
 915F   D0 F4       BNE   $9155    ;nein, dann Ende u. Fehler
 9161   AC 1D C5    LDY   ERRLN    ;Zeilennummer, wo Fehler
 9164   AE 1E C5    LDX   ERRLN+1  ;auftrat in ERRLN,ERRLN+1
 9167   20 DB 83    JSR   INCBASBZ ;Code ("N") überlesen
 916A   4C CD 8C    JMP   SXYFLP   ;Zeilennnr. n. Fliesskomma

FNERRN:
 916D   AC FC C5    LDY   ERRN     ;Fehlernr. steht in ERRN
 9170   A2 00       LDX   #$00     ;High-Byte = 0
 9172   F0 F3       BEQ   $9167    ;weiter wie oben (unbed.)

BEFO:                              ;Leerbefehl (z.B. DISAPA)
 9174   20 73 00    JSR   CHRGET   ;Code überlesen
 9177   4C 2A 82    JMP   ENDSMB   ;Simon's-Befehl beenden

SBASBEF:                           ;einen Befehl ausfuehren
 917A   A5 39       LDA   $39      ;aktuelle Zeilennummer
 917C   CD 86 C5    CMP   ZN       ;vorige Zeilennr.
 917F   D0 07       BNE   $9188    ;ungleich, dann weiter
 9181   A5 3A       LDA   $3A
 9183   CD 87 C5    CMP   ZN+1     ;Zeilennummern gleich ?
 9186   F0 1E       BEQ   $91A6    ;ja, dann zu $91A6
 9188   A5 39       LDA   $39      ;vorige Znr. = akt. Znr.
 918A   8D 86 C5    STA   ZN
 918D   A5 3A       LDA   $3A
 918F   8D 87 C5    STA   ZN+1
 9192   AD 01 C6    LDA   TRACEFLAG ;TRACE aktiv ?
 9195   C9 0A       CMP   #$0A
 9197   D0 03       BNE   $919C     ;nein ?
 9199   20 6A 89    JSR   TRACE1    ;TRACE behandeln
 919C   AD EA C5    LDA   ONKEYFLAG ;ONKEY aktiv ?
 919F   C9 0A       CMP   #$0A
 91A1   D0 03       BNE   $91A6     ;nein ?
 91A3   20 BA 9D    JSR   ONKEY1    ;ONKEY behandeln
 91A6   20 73 00    JSR   CHRGET    ;Basic-Code holen
 91A9   A9 00       LDA   #$00
 91AB   8D 52 CA    STA   FLAG1     ;Flags annullieren
 91AE   8D B2 C5    STA   AUTOFLAG
 91B1   AD 01 C6    LDA   TRACEFLAG ;Trace aktiv ?
 91B4   C9 0A       CMP   #$0A
 91B6   D0 03       BNE   $91BB     ;nein ?
 91B8   20 E2 89    JSR   TRACESHOW ;Trace-Fenster malen
 91BB   A0 00       LDY   #$00
 91BD   B1 7A       LDA   ($7A),Y   ;Basic-Code
 91BF   C9 8B       CMP   #$8B      ;Code für IF
 91C1   D0 03       BNE   $91C6     ;nein ?
 91C3   4C 6C 9B    JMP   BEFIF     ;zum IF-Befehl
```

```
 91C6   C9 64       CMP  #$64      ;Praefix für Simon's-Bef.
 91C8   F0 20       BEQ  $91EA     ;ja, dann zum Sprungvert.
 91CA   C9 8A       CMP  #$8A      ;Code für RUN
 91CC   F0 0E       BEQ  $91DC
 91CE   C9 9C       CMP  #$9C      ;Code für CLR
 91D0   F0 0A       BEQ  $91DC
 91D2   C9 A2       CMP  #$A2      ;Code für NEW
 91D4   F0 06       BEQ  $91DC
 91D6   20 79 00    JSR  CHRGOT    ;Code holen
 91D9   4C E7 A7    JMP  $A7E7     ;zur normalen Basic-
                                   ;Befehls-Dekodier-Routine
 91DC   A2 00       LDX  #$00
 91DE   8E 17 C6    STX  SPREPEAT  ;Stacks initialisieren
 91E1   8E 2C C6    STX  SPEXEC
 91E4   8E 41 C6    STX  SPLOOP
 91E7   4C D6 91    JMP  $91D6     ;weiter wie oben

 91EA   20 DB 83    JSR  INCBASBZ  ;Code ($64) überlesen
 91ED   20 EC 82    JSR  BASROMAU  ;Basic aus
 91F0   B1 7A       LDA  ($7A),Y
 91F2   0A          ASL  A         ;Code verdoppeln
 91F3   A8          TAY
 91F4   88          DEY
 91F5   88          DEY        ;minus 2; als Zeiger in SPRTAB
 91F6   B9 C5 86    LDA  SPRTAB+1,Y ;high-Byte (Zieladr.-1)
 91F9   48          PHA            ;auf Stapel
 91FA   B9 C4 86    LDA  SPRTAB,Y  ;low-Byte (Zieladr.-1)
 91FD   48          PHA            ;auf Stapel
 91FE   60          RTS            ;Sprung zu Zieladr.

BEFHIRES:
 91FF   A9 01       LDA  #$01      ;XMAX auf 319 setzen
 9201   8D 2E C5    STA  XMAXHIGH
 9204   A9 3F       LDA  #$3F
 9206   8D 2D C5    STA  XMAXLOW
 9209   A9 00       LDA  #$00
 920B   85 A6       STA  $A6
 920D   A0 00       LDY  #$00
 920F   8C 1D CB    STY  LOWCOLFLAG ;annullieren
 9212   A9 E0       LDA  #$E0
 9214   85 A7       STA  $A7       ;($A6,$A7) auf $E000
 9216   20 CC 99    JSR  GET2NYB   ;2 Nybbles (Farben) holen
 9219   A6 AA       LDX  $AA
 921B   86 AC       STX  $AC       ;akt. Farben
 921D   A9 00       LDA  #$00
 921F   91 A6       STA  ($A6),Y   ;Grafik-RAM loeschen
 9221   20 E9 9A    JSR  INCA6A7   ;nächste Zelle
 9224   A5 A6       LDA  $A6
 9226   C9 40       CMP  #$40      ;Ende low erreicht ?
 9228   D0 F3       BNE  $921D     ;nein, dann Schleife
 922A   A5 A7       LDA  $A7
```

```
 922C   C9 FF       CMP  #$FF      ;Ende high erreicht ?
 922E   D0 ED       BNE  $921D     ;nein, dann Schleife
 9230   A9 00       LDA  #$00
 9232   8D B0 C5    STA  MULTIJN   ;Multi-Flag loeschen
 9235   20 37 A8    JSR  NRM       ;Speicherverteilung norm.
 9238   A9 0A       LDA  #$0A
 923A   8D B3 C5    STA  GMEMFLAG  ;Flag entsprechend setzen
 923D   A9 3B       LDA  #$3B
 923F   8D 11 D0    STA  VICST1    ;VIC Steuerreg. 1
 9242   A9 0B       LDA  #$0B
 9244   8D 18 D0    STA  VICADS    ;VIC Adressregister
 9247   A9 94       LDA  #$94
 9249   8D 00 DD    STA  CIA2PRA   ;CIA2-Port (stellt Bit 14
                                   ;und 15 v. Grafik RAM her)
 924C   A0 00       LDY  #$00
 924E   8C 1D CB    STY  LOWCOLFLAG
 9251   A5 AC       LDA  $AC       ;aktuelle Farbkombination
 9253   99 00 C0    STA  $C000,Y   ;in Video-RAM schreiben
 9256   99 FF C0    STA  $C0FF,Y
 9259   99 FE C1    STA  $C1FE,Y
 925C   99 FD C2    STA  $C2FD,Y
 925F   C8          INY
 9260   C0 FF       CPY  #$FF
 9262   D0 ED       BNE  $9251     ;Schleife f. Video-RAM
 9264   4C 2A 82    JMP  ENDSMB    ;Befehl beenden

BEFPLOT:
 9267   20 DB 83    JSR  INCBASBZ  ;Code ueberlesen
 926A   20 27 94    JSR  SGETADR   ;Wert (X-Koord.) holen
 926D   84 09       STY  $09       ;nach $09,$0A
 926F   85 0A       STA  $0A
 9271   20 FC 81    JSR  SGETBYTC  ;Byte-Wert (Y) holen
 9274   86 A4       STX  $A4       ;nach $A4
 9276   20 7D 93    JSR  GETPKTF   ;Punktfarbe holen
 9279   A5 0A       LDA  $0A
 927B   F0 0F       BEQ  $928C     ;X kleiner 256, dann ok
 927D   A5 09       LDA  $09
 927F   C9 7F       CMP  #$7F
 9281   D0 09       BNE  $928C     ;X ungleich 383, dann ok
 9283   A5 F7       LDA  $F7       ;Punktfarbe
 9285   C9 7F       CMP  #$7F
 9287   D0 03       BNE  $928C     ;ungleich 127, dann ok

 9289   4C AA BD    JMP  DSCBMOUT1 ;Sprung nach $9A5A

 928C   20 97 92    JSR  PUNKT     ;Punkt setzen
 928F   4C 2A 82    JMP  ENDSMB    ;Befehl beenden

 9292   A9 08       LDA  #$08
 9294   85 90       STA  $90
 9296   60          RTS
```

```
PUNKT:                    ;X in $09,$0A; Y in $A4; Farbe in $F7
 9297  18         CLC
 9298  A5 09      LDA  $09
 929A  ED 2D C5   SBC  XMAXLOW  ;Plausibilitaet X-Koord.
 929D  A5 0A      LDA  $0A
 929F  ED 2E C5   SBC  XMAXHIGH
 92A2  B0 EE      BCS  $9292    ;X zu gross, dann ST=8
 92A4  A5 A4      LDA  $A4
 92A6  C9 C8      CMP  #$C8     ;Plausibilitaet Y-Koord.
 92A8  B0 E8      BCS  $9292    ;Y zu gross, dann ST=8
 92AA  AD B0 C5   LDA  MULTIJN  ;Multi-Colour-Mode ?
 92AD  F0 04      BEQ  $92B3    ;nein, dann weiter
 92AF  06 09      ASL  $09      ;X verdoppeln
 92B1  26 0A      ROL  $0A
 92B3  A5 09      LDA  $09
 92B5  29 07      AND  #$07     ;(X and 7)
 92B7  AA         TAX           ;X-Reg. = Stelle im Byte
 92B8  A5 A4      LDA  $A4
 92BA  29 07      AND  #$07
 92BC  85 63      STA  $63      ;(Y and 7) nach $63
 92BE  A5 09      LDA  $09
 92C0  29 F8      AND  #$F8     ;X = (X and $FFF8)
 92C2  85 09      STA  $09
 92C4  A5 F7      LDA  $F7      ;Punktfarbe
 92C6  C9 0B      CMP  #$0B     ;$0B fuer TEST
 92C8  F0 1F      BEQ  $92E9    ;ja, dann weiter
 92CA  AD 1D CB   LDA  LOWCOLFLAG
 92CD  F0 1A      BEQ  $92E9 ;LOWCOL inaktiv, dann weiter
 92CF  AD AE C5   LDA  GFLAG
 92D2  D0 15      BNE  $92E9 ;GFLAG gestezt, dann weiter
 92D4  A5 A4      LDA  $A4   ;Y und X retten
 92D6  48         PHA
 92D7  A5 09      LDA  $09
 92D9  48         PHA
 92DA  A5 0A      LDA  $0A
 92DC  48         PHA
 92DD  20 83 93   JSR  FARBSET ;Farbe lokal aendern
 92E0  68         PLA          ;Y und X wiederherstellen
 92E1  85 0A      STA  $0A
 92E3  68         PLA
 92E4  85 09      STA  $09
 92E6  68         PLA
 92E7  85 A4      STA  $A4
 92E9  A5 A4      LDA  $A4     ;Y-Koord.
 92EB  4A         LSR  A
 92EC  4A         LSR  A
 92ED  4A         LSR  A       ;Y div 8
 92EE  A8         TAY
 92EF  18         CLC
 92F0  B9 2D 94   LDA  GADRTABL,Y ;Tab.: $E000+320*Y
 92F3  65 63      ADC  $63     ;Offset von (Y and 7)
```

```
92F5   65 09       ADC   $09         ;Offset von X-Koord.
92F7   85 A8       STA   $A8         ;gibt Adresse low
92F9   B9 46 94    LDA   GADRTABH,Y ;analog High-Bytes
92FC   65 0A       ADC   $0A
92FE   85 A9       STA   $A9
9300   A0 00       LDY   #$00        ;Y-Reg. annullieren
9302   84 90       STY   $90         ;ST=0 (Statusbyte)
9304   78          SEI
9305   A5 01       LDA   $01
9307   29 FD       AND   #$FD        ;Kernal-ROM ausschalten
9309   85 01       STA   $01
930B   AD AE C5    LDA   GFLAG
930E   F0 01       BEQ   $9311       ;wenn GFLAG=0, dann weiter
9310   60          RTS          ;GFLAG=$FF beim Blockzeichn.

9311   AD B0 C5    LDA   MULTIJN
9314   F0 03       BEQ   $9319       ;kein Multi, dann weiter
9316   4C 7D 9A    JMP   $9A7D       ;Punkt im MCM

Punkt im normalen Grafik-Modus
9319   A5 F7       LDA   $F7         ;Punktfarbe
931B   F0 1B       BEQ   $9338       ;Punkt loeschen bei $9338
931D   C9 01       CMP   #$01
931F   F0 07       BEQ   $9328       ;Punkt setzen bei $9328
9321   C9 02       CMP   #$02
9323   F0 23       BEQ   $9348       ;Punkt invert. bei $9348
9325   4C 61 A2    JMP   PTEST1      ;Punkt testen

9328   B1 A8       LDA   ($A8),Y
932A   1D 30 93    ORA   GBITTABS,X ;entspr. Bit setzen
932D   91 A8       STA   ($A8),Y
932F   60          RTS               ;fertig

GBITTABS:
9330   80 40 20 10 08 04 02 01   ;Tabelle f. Punkt setzen

9338   B1 A8       LDA   ($A8),Y
933A   3D 40 93    AND   GBITTABL,X ;entspr. Bit loeschen
933D   91 A8       STA   ($A8),Y
933F   60          RTS

GBITTABL:
9340   7F BF DF EF F7 FB FD FE   ;Tabelle f. Punkt loeschen

9348   B1 A8       LDA   ($A8),Y
934A   5D 30 93    EOR   GBITTABS,X ;entspr. Bit invert.
934D   91 A8       STA   ($A8),Y
934F   60          RTS

BEFLINE:
9350   20 59 93    JSR   GETXYXYF ;AX,AY,EX,EY und Farbe
```

```
 9353   20 5E A3    JSR   LINZEI    ;Linie zeichnen
 9356   4C 2A 82    JMP   ENDSMB    ;Befehl beenden

GETXYXYF:
 9359   20 DB 83    JSR   INCBASBZ ;Code ueberlesen
 935C   20 27 94    JSR   SGETADR   ;Wert f. AX holen
 935F   8D 98 C5    STA   AX+1
 9362   8C 97 C5    STY   AX
 9365   20 FC 81    JSR   SGETBYTC ;Wert f. AY holen
 9368   8E AC C5    STX   AY
 936B   20 31 80    JSR   SCHKCOM   ;Komma ?
 936E   20 27 94    JSR   SGETADR   ;Wert f. EX holen
 9371   8D AB C5    STA   EX+1
 9374   8C AA C5    STY   EX
 9377   20 FC 81    JSR   SGETBYTC ;Wert f. EY holen
 937A   8E A8 C5    STX   EY

GETPKTF:
 937D   20 FC 81    JSR   SGETBYTC ;Wert f. Pkt.Farbe holen
 9380   86 F7       STX   $F7       ;in $F7 speichern
 9382   60          RTS

FARBSET:                             ;Farbe lokal aendern
 9383   20 0E 94    JSR   KERROMEIN ;Kernal einschalten
 9386   A5 09       LDA   $09
 9388   4A          LSR   A
 9389   66 0A       ROR   $0A
 938B   66 09       ROR   $09
 938D   A5 09       LDA   $09
 938F   4A          LSR   A
 9390   66 0A       ROR   $0A
 9392   66 09       ROR   $09
 9394   A5 09       LDA   $09
 9396   4A          LSR   A
 9397   66 0A       ROR   $0A
 9399   66 09       ROR   $09       ;$09,$0A 3 Bit rechts rot.
 939B   46 A4       LSR   $A4
 939D   46 A4       LSR   $A4
 939F   46 A4       LSR   $A4       ;$A4 = ($A4 div 8)
 93A1   20 5F 94    JSR   A4MAL40   ;$A4,$A5 = $A4 * 40
 93A4   18          CLC
 93A5   A5 A4       LDA   $A4
 93A7   65 09       ADC   $09
 93A9   85 A8       STA   $A8
 93AB   85 AA       STA   $AA
 93AD   A5 A5       LDA   $A5
 93AF   65 0A       ADC   $0A
 93B1   85 A9       STA   $A9    ;$A8,$A9 = $A4,$A5 + $09,$0A
 93B3   85 AB       STA   $AB    ;$AA,$AB = $A4,$A5 + $09,$0A
 93B5   18          CLC
 93B6   A5 A8       LDA   $A8
```

```
 93B8   69 00        ADC  #$00
 93BA   85 A8        STA  $A8
 93BC   A5 A9        LDA  $A9
 93BE   69 C0        ADC  #$C0
 93C0   85 A9        STA  $A9        ;$A8,$A9 = $A8,$A9 + $C000
 93C2   A0 00        LDY  #$00
 93C4   AD 1B CB     LDA  LCFARB12   ;Lowcol-Farben 1 & 2
 93C7   91 A8        STA  ($A8),Y    ;in Video-RAM
 93C9   18           CLC
 93CA   A5 AA        LDA  $AA
 93CC   69 00        ADC  #$00
 93CE   85 A8        STA  $A8
 93D0   A5 AB        LDA  $AB
 93D2   69 D8        ADC  #$D8
 93D4   85 A9        STA  $A9        ;$A8,$A9 = $AA,$AB + $D800
 93D6   AD 1C CB     LDA  LCFARB3    ;Lowcol-Farbe 3
 93D9   91 A8        STA  ($A8),Y    ;in Farb-RAM
 93DB   60           RTS

BEFLOWCOL:
 93DC   20 02 82     JSR  SGETBYTN ;1.Farbe holen
 93DF   8E 1B CB     STX  LCFARB12
 93E2   0E 1B CB     ASL  LCFARB12
 93E5   0E 1B CB     ASL  LCFARB12
 93E8   0E 1B CB     ASL  LCFARB12
 93EB   0E 1B CB     ASL  LCFARB12 ;mal 16
 93EE   20 FC 81     JSR  SGETBYTC ;2.Farbe holen
 93F1   8A           TXA
 93F2   0D 1B CB     ORA  LCFARB12 ;mit altem Wert oderieren
 93F5   8D 1B CB     STA  LCFARB12 ;gibt Farbkombination 1&2
 93F8   A9 0A        LDA  #$0A
 93FA   8D 1D CB     STA  LOWCOLFLAG ;LOWCOL-Flag setzen
 93FD   20 FC 81     JSR  SGETBYTC ;3.Farbe holen
 9400   8E 1C CB     STX  LCFARB3  ;speichern
 9403   4C 2A 82     JMP  ENDSMB   ;Befehl fertig

KERROMAUS:
 9406   78           SEI           ;Interrupts verhindern
 9407   A5 01        LDA  $01      ;Prozessor-Port
 9409   29 FD        AND  #$FD     ;Bit 1 annullieren
 940B   85 01        STA  $01
 940D   60           RTS

KERROMEIN:
 940E   A5 01        LDA  $01
 9410   09 02        ORA  #$02     ;Bit 1 setzen
 9412   85 01        STA  $01
 9414   60           RTS

SGETADR2:             ;(unnoetig) umrahmter Aufruf von GETADR
 9415   AD 96 C5     LDA  PORTSPEI
```

```
 9418   48          PHA
 9419   20 27 94    JSR   SGETADR
 941C   8D 7D C5    STA   ZWSPEI1
 941F   68          PLA
 9420   8D 96 C5    STA   PORTSPEI
 9423   AD 7D C5    LDA   ZWSPEI1
 9426   60          RTS

SGETADR:
 9427   20 0D 80    JSR   SFRMNUM ;Term holen
 942A   4C 16 80    JMP   SFACADR ;in Adressformat wandeln

GADRTABL:  ;Tabelle fuer Low-Bytes von $E000 + 320 * Index
 942D   00 40 80 C0 00 40 80 C0
 9435   00 40 80 C0 00 40 80 C0
 943D   00 40 80 C0 00 40 80 C0
 9445   00

GADRTABH:  ;Tabelle f. High-Bytes von $E000 + 320 * Index
 9446   E0 E1 E2 E3 E5 E6 E7 E8
 944E   EA EB EC ED EF F0 F1 F2
 9456   F4 F5 F6 F7 F9 FA FB FC
 945E   FE

A4MAL40:
 945F   A9 00       LDA   #$00
 9461   85 A5       STA   $A5       ;$A5=0
 9463   06 A4       ASL   $A4
 9465   26 A5       ROL   $A5
 9467   06 A4       ASL   $A4
 9469   26 A5       ROL   $A5
 946B   06 A4       ASL   $A4
 946D   26 A5       ROL   $A5       ;$A4,$A5 = $A4,$A5 * 8
 946F   A5 A5       LDA   $A5
 9471   48          PHA
 9472   A5 A4       LDA   $A4
 9474   48          PHA
 9475   06 A4       ASL   $A4
 9477   26 A5       ROL   $A5
 9479   06 A4       ASL   $A4
 947B   26 A5       ROL   $A5       ;$A4,$A5 enth. Wert * 32
 947D   18          CLC
 947E   68          PLA
 947F   65 A4       ADC   $A4
 9481   85 A4       STA   $A4
 9483   68          PLA
 9484   65 A5       ADC   $A5
 9486   85 A5       STA   $A5       ;Wert * 40
 9488   60          RTS
```

```
BEFCIRCLE:
 9489  20 DB 83   JSR   INCBASBZ ;Code ueberlesen
 948C  A9 00      LDA   #$00
 948E  8D 7C C5   STA   DREHSINN
 9491  8D 2F C5   STA   UEBERDREH
 9494  20 27 94   JSR   SGETADR  ;Mittelpkt. X-Koord. holen
 9497  8C 4B CB   STY   KMX
 949A  8D 4C CB   STA   KMX+1
 949D  20 FC 81   JSR   SGETBYTC ;Mittelpkt. Y-Koord. holen
 94A0  8E 49 CB   STX   KMY
 94A3  A9 00      LDA   #$00
 94A5  85 A8      STA   $A8      ;Startwinkel
 94A7  85 A9      STA   $A9
 94A9  8D 4E CB   STA   WINKSW+1 ;Winkelschritt (High)
 94AC  A9 68      LDA   #$68
 94AE  85 AC      STA   $AC
 94B0  A9 01      LDA   #$01
 94B2  85 AD      STA   $AD      ;$AC,$AD = 360 (Endwinkel)
 94B4  A9 0C      LDA   #$0C
 94B6  8D 4D CB   STA   WINKSW   ;Winkelschrittweite = 12
 94B9  20 FC 81   JSR   SGETBYTC ;Radius X holen
 94BC  8E 48 CB   STX   KRX
 94BF  20 FC 81   JSR   SGETBYTC ;Radius Y holen
 94C2  8E C5 CB   STX   KRY
 94C5  20 7D 93   JSR   GETPKTF  ;Punktfarbe
 94C8  20 14 A9   JSR   KREIS    ;Kreis zeichnen
 94CB  4C 2A 82   JMP   ENDSMB   ;Befehl fertig

20PL40:
 94CE  18         CLC
 94CF  A5 20      LDA   $20
 94D1  69 28      ADC   #$28     ;+40
 94D3  85 20      STA   $20
 94D5  A5 21      LDA   $21
 94D7  69 00      ADC   #$00
 94D9  85 21      STA   $21      ;$20,$21 = $20,$21 + 40
 94DB  60         RTS

23PL40:
 94DC  18         CLC
 94DD  A5 23      LDA   $23
 94DF  69 28      ADC   #$28     ;+40
 94E1  85 23      STA   $23
 94E3  A5 24      LDA   $24
 94E5  69 00      ADC   #$00
 94E7  85 24      STA   $24      ;$23,$24 = $23,$24 + 40
 94E9  60         RTS

BEFMULTI:
 94EA  A9 2C      LDA   #$2C
 94EC  8D B0 C5   STA   MULTIJN  ;Multi-Flag setzen
```

```
 94EF   A9 9F      LDA  #$9F
 94F1   8D 2D C5   STA  XMAXLOW
 94F4   A9 00      LDA  #$00
 94F6   8D 2E C5   STA  XMAXHIGH ;XMAX = 159
 94F9   20 CC 99   JSR  GET2NYB  ;2 Werte holen
 94FC   A0 00      LDY  #$00
 94FE   A2 00      LDX  #$00
 9500   8A         TXA
 9501   A5 AA      LDA  $AA      ;Farbkombination 1&2
 9503   99 00 C0   STA  $C000,Y  ;in Video-RAM
 9506   99 F9 C0   STA  $C0F9,Y
 9509   99 F3 C1   STA  $C1F3,Y
 950C   99 ED C2   STA  $C2ED,Y
 950F   E8         INX
 9510   C8         INY
 9511   C0 FB      CPY  #$FB
 9513   D0 EB      BNE  $9500    ;Schleife
 9515   20 FC 81   JSR  SGETBYTC ;3.Farbe holen
 9518   86 AA      STX  $AA
 951A   A0 00      LDY  #$00
 951C   A5 AA      LDA  $AA
 951E   99 00 D8   STA  $D800,Y  ;in Farb-RAM
 9521   99 F9 D8   STA  $D8F9,Y
 9524   99 F3 D9   STA  $D9F3,Y
 9527   99 ED DA   STA  $DAED,Y
 952A   C8         INY
 952B   C0 FB      CPY  #$FB
 952D   D0 ED      BNE  $951C    ;Schleife
 952F   A9 18      LDA  #$18
 9531   8D 16 D0   STA  VICST2   ;VIC Steuerregister 2
 9534   4C 2A 82   JMP  ENDSMB   ;Befehl fertig

BEFCOLOUR:
 9537   20 02 82   JSR  SGETBYTN ;Wert holen
 953A   8E 20 D0   STX  VICRAFAR ;Rahmenfarbe
 953D   20 FC 81   JSR  SGETBYTC ;Wert holen
 9540   8E 21 D0   STX  VICHIFAR ;Hintergrundfarbe
 9543   4C 2A 82   JMP  ENDSMB   ;Befehl fertig

BEFDIR:
 9546   20 DB 83   JSR  INCBASBZ ;Code ueberlesen
 9549   20 CC FF   JSR  CLRCH    ;Kanaele schliessen
 954C   A9 01      LDA  #$01
 954E   20 C3 FF   JSR  CLOSE    ;Datei #1 schliessen
 9551   20 CC FF   JSR  CLRCH    ;Kanaele schliessen
 9554   20 B7 86   JSR  SGETSTR  ;String holen
 9557   86 AE      STX  $AE      ;Adresse nach $AE,$AF
 9559   85 AF      STA  $AF
 955B   A0 00      LDY  #$00
 955D   B1 AE      LDA  ($AE),Y  ;Zeichen aus String
 955F   C9 24      CMP  #$24     ;"$"
```

```
9561  F0 05     BEQ  $9568     ;ja, dann weiter
9563  A9 00     LDA  #$00      ;Nr.f. 'bad mode'
9565  4C 8C 88  JMP  SERROUT   ;Fehlermeldung ausgeben

9568  A4 AF     LDY  $AF
956A  A5 69     LDA  $69
956C  20 BD FF  JSR  SETFNPAR  ;Filenamenparameter setzen
956F  A9 01     LDA  #$01      ;File #1
9571  A2 08     LDX  #$08      ;Geraetenr. 8
9573  A0 00     LDY  #$00      ;Sekundaeradresse 0
9575  20 BA FF  JSR  SETFPAR   ;Fileparameter setzen
9578  20 C0 FF  JSR  OPEN      ;Datei oeffnen
957B  A2 01     LDX  #$01      ;Datei #1
957D  20 C6 FF  JSR  CHKIN     ;als Eingabekanal
9580  20 CF FF  JSR  BASIN     ;'Startadresse' ueberlesen
9583  20 CF FF  JSR  BASIN
9586  20 CF FF  JSR  BASIN     ;'Vorwaertszeiger' ueberl.
9589  20 CF FF  JSR  BASIN
958C  20 CF FF  JSR  BASIN     ;Low-Byte Blockzahl
958F  85 63     STA  $63
9591  20 CF FF  JSR  BASIN     ;High-Byte Blockzahl
9594  85 62     STA  $62
9596  20 4C 80  JSR  SINTOUT   ;Integer-Wert ausgeben
9599  20 CF FF  JSR  BASIN     ;Zeichen Dateiname
959C  A6 90     LDX  $90       ;Statusbyte ST
959E  F0 0F     BEQ  $95AF     ;=0, dann weiter
95A0  20 CC FF  JSR  CLRCH     ;Kanaele schliessen
95A3  A9 01     LDA  #$01
95A5  20 C3 FF  JSR  CLOSE     ;Datei #1 schliessen
95A8  20 CC FF  JSR  CLRCH     ;Kanaele schliessen
95AB  58        CLI
95AC  4C 2A 82  JMP  ENDSMB    ;Befehl fertig

95AF  20 D2 FF  JSR  BSOUT     ;Zeichen ausgeben
95B2  C9 22     CMP  #$22      ;Anfuehrungszeichen
95B4  D0 E3     BNE  $9599     ;nein, dann Schleife
95B6  20 CF FF  JSR  BASIN     ;naechstes Zeichen
95B9  20 D2 FF  JSR  BSOUT     ;ausgeben
95BC  C9 00     CMP  #$00      ;Zeilenende
95BE  D0 F6     BNE  $95B6     ;nein, dann Schleife
95C0  A9 0D     LDA  #$0D      ;Carriage Return
95C2  20 D2 FF  JSR  BSOUT     ;ausgeben
95C5  4C 86 95  JMP  $9586     ;Schleife

BEFMMOB:
95C8  20 02 82  JSR  SGETBYTN  ;Sprite-Nr. holen
95CB  8E BB C5  STX  MOBNR2
95CE  8E BC C5  STX  MOBNR
95D1  0E BB C5  ASL  MOBNR2    ;in MOBNR2 doppelte Nr.
95D4  20 31 80  JSR  SCHKCOM   ;Komma ?
95D7  A5 7A     LDA  $7A
```

```
 95D9  D0 02     BNE  $95DD
 95DB  C6 7B     DEC  $7B
 95DD  C6 7A     DEC  $7A   ;Basic-Zeiger um 1 vermindert
 95DF  20 59 93  JSR  GETXYXYF ;AX,AY,EX,EY und $F7 holen
 95E2  A5 F7     LDA  $F7      ;Vergroesserungsbyte
 95E4  D0 09     BNE  $95EF    ;nicht null, dann weiter
 95E6  20 5B 8C  JSR  MOBEXL   ;Vergr. X-Ri. loeschen
 95E9  20 68 8C  JSR  MOBEYL   ;Vergr. Y-Ri. loeschen
 95EC  4C 0F 96  JMP  $960F

 95EF  C9 01     CMP  #$01
 95F1  D0 09     BNE  $95FC    ;nicht 1, dann weiter
 95F3  20 41 8C  JSR  MOBEXS   ;Vergr. X-Ri. setzen
 95F6  20 68 8C  JSR  MOBEYL   ;Vergr. Y-Ri. loeschen
 95F9  4C 0F 96  JMP  $960F

 95FC  C9 02     CMP  #$02
 95FE  D0 09     BNE  $9609    ;nicht 2, dann weiter
 9600  20 5B 8C  JSR  MOBEXL   ;Vergr. X-Ri. loeschen
 9603  20 4E 8C  JSR  MOBEYS   ;Vergr. Y-Ri. setzen
 9606  4C 0F 96  JMP  $960F

 9609  20 41 8C  JSR  MOBEXS   ;Vergr. X-Ri. setzen
 960C  20 4E 8C  JSR  MOBEYS   ;Vergr. Y-Ri. setzen

 960F  20 FC 81  JSR  SGETBYTC ;speed holen
 9612  8E B5 C5  STX  MOBSPEED
 9615  A9 F0     LDA  #$F0
 9617  8D B4 C5  STA  MOBBEW   ;Flag setzen
 961A  20 AC A3  JSR  MOBBEWEG ;Sprite bewegen
 961D  A9 00     LDA  #$00
 961F  8D B4 C5  STA  MOBBEW   ;Flag loeschen
 9622  4C 2A 82  JMP  ENDSMB   ;Befehl fertig

MOBPOSS:
 9625  A5 09     LDA  $09      ;neue X-Position Low
 9627  A2 00     LDX  #$00
 9629  EC B5 C5  CPX  MOBSPEED ;Verzoegerungsschleife
 962C  F0 0B     BEQ  $9639    ;fertig, dann weiter
 962E  E8        INX           ;auesseren Zaehler
 962F  A0 00     LDY  #$00
 9631  C0 0A     CPY  #$0A
 9633  F0 F4     BEQ  $9629    ;auessere Schleife
 9635  C8        INY           ;innerer Zaehler
 9636  4C 31 96  JMP  $9631    ;innere Schleife

 9639  AC BB C5  LDY  MOBNR2   ;Sprite-Nr. * 2
 963C  99 00 D0  STA  VICMX,Y  ;X-Position setzen
 963F  A5 0A     LDA  $0A      ;X-Position high
 9641  C9 01     CMP  #$01
 9643  D0 15     BNE  $965A    ;nicht 1, dann weiter
```

```
 9645   AC BC C5   LDY   MOBNR
 9648   AD 10 D0   LDA   VICMX8
 964B   19 78 97   ORA   MBITTABS,Y ;X8-Bit setzen
 964E   8D 10 D0   STA   VICMX8      ;X-Uebertragsregister
 9651   A5 A4      LDA   $A4        ;neue Y-Koordinate
 9653   AC BB C5   LDY   MOBNR2
 9656   99 01 D0   STA   VICMY,Y    ;Y-Koord. setzen
 9659   60         RTS

 965A   AC BC C5   LDY   MOBNR
 965D   AD 10 D0   LDA   VICMX8
 9660   39 80 97   AND   MBITTABL,Y ;X8-Bit loeschen
 9663   8D 10 D0   STA   VICMX8
 9666   4C 51 96   JMP   $9651       ;weiter wie oben

BEFBFLASH:
 9669   A9 0A      LDA   #$0A
 966B   8D DB C5   STA   BFLASHJN ;BFLASH-Flag setzen
 966E   20 02 82   JSR   SGETBYTN ;speed holen
 9671   E0 00      CPX   #$00
 9673   D0 08      BNE   $967D    ;nicht null, dann weiter
 9675   A9 00      LDA   #$00
 9677   8D DB C5   STA   BFLASHJN ;Flag loeschen
 967A   4C 2A 82   JMP   ENDSMB   ;Befehl fertig

 967D   8E D7 C5   STX   BFLASHSP ;speed setzen
 9680   20 FC 81   JSR   SGETBYTC ;1.Farbe holen
 9683   8E D9 C5   STX   BFLASHF1 ;speichern
 9686   20 FC 81   JSR   SGETBYTC ;2.Farbe holen
 9689   8E DA C5   STX   BFLASHF2 ;speichern
 968C   A9 00      LDA   #$00
 968E   8D D8 C5   STA   BFLZAE   ;Zaehler = 0
 9691   4C 2A 82   JMP   ENDSMB   ;Befehl fertig

SIRQ:                             ;IRQ-Interrupt-Routine
 9694   EE 16 C5   INC   ZAEIRQ   ;Zaehler erhoehen
 9697   AD 16 C5   LDA   ZAEIRQ
 969A   C9 3C      CMP   #$3C     ;60
 969C   D0 08      BNE   $96A6
 969E   EE 17 C5   INC   ZAESEC   ;Sekundenzaehler erhoehen
 96A1   A9 00      LDA   #$00
 96A3   8D 16 C5   STA   ZAEIRQ   ;Zaehler annullieren
 96A6   AD DB C5   LDA   BFLASHJN ;BFLASH aktiv ?
 96A9   C9 0A      CMP   #$0A
 96AB   D0 2A      BNE   $96D7    ;nein, dann weiter
 96AD   EE D8 C5   INC   BFLZAE   ;Zaehler erhoehen
 96B0   AD D8 C5   LDA   BFLZAE
 96B3   CD D7 C5   CMP   BFLASHSP ;gleich Speedwert ?
 96B6   D0 1F      BNE   $96D7    ;nein, dann weiter
 96B8   A9 00      LDA   #$00
 96BA   8D D8 C5   STA   BFLZAE   ;Zaehler = 0
```

```
 96BD   38          SEC
 96BE   AD 20 D0    LDA   VICRAFAR ;akt. Rahmenfarbe
 96C1   E9 F0       SBC   #$F0
 96C3   CD D9 C5    CMP   BFLASHF1 ;gleich 1.Farbe ?
 96C6   F0 09       BEQ   $96D1    ;ja, dann weiter
 96C8   AD D9 C5    LDA   BFLASHF1 ;1. Farbe
 96CB   8D 20 D0    STA   VICRAFAR ;setzen
 96CE   4C D7 96    JMP   $96D7
 96D1   AD DA C5    LDA   BFLASHF2 ;2.Farbe
 96D4   8D 20 D0    STA   VICRAFAR ;setzen
 96D7   4C D2 97    JMP   SIRQ2    ;Fortsetzung

BEFMOBSET:
 96DA   20 02 82    JSR   SGETBYTN ;Sprite-Nr. holen
 96DD   86 AA       STX   $AA      ;in $AA speichern
 96DF   AD 15 D0    LDA   VICMEA   ;Sprite-Ein/Aus-Reg.
 96E2   1D 78 97    ORA   MBITTABS,X ;Bit setzen
 96E5   8D 15 D0    STA   VICMEA
 96E8   20 FC 81    JSR   SGETBYTC ;Blocknr. holen
 96EB   A4 AA       LDY   $AA      ;Sprite-Nr.
 96ED   AD 88 02    LDA   VIDRAMHI ;Video-RAM Lage
 96F0   C9 CC       CMP   #$CC     ;$CC (nach MEM)
 96F2   F0 0E       BEQ   $9702    ;ja, dann zu $9702
 96F4   AD B3 C5    LDA   GMEMFLAG ;Flag bei Hires
 96F7   C9 0A       CMP   #$0A
 96F9   D0 0E       BNE   $9709    ;nein, dann weiter
 96FB   8A          TXA            ;Blocknr.
 96FC   99 F8 C3    STA   $C3F8,Y  ;Blockregister, wenn
                                   ;Video-RAM bei $C000
 96FF   4C 0D 97    JMP   $970D

 9702   8A          TXA            ;Blocknr.
 9703   99 F8 CF    STA   $CFF8,Y  ;Reg,wenn VidRAM b. $CC00
 9706   4C 0D 97    JMP   $970D

 9709   8A          TXA            ;Blocknr.
 970A   99 F8 07    STA   $07F8,Y  ;Reg,wenn VidRAM b. $0400

 970D   20 FC 81    JSR   SGETBYTC ;Farbe holen
 9710   A4 AA       LDY   $AA      ;Spritenr.
 9712   8A          TXA
 9713   99 27 D0    STA   VICMFA,Y ;Farbe setzen
 9716   20 FC 81    JSR   SGETBYTC ;Prioritaet holen
 9719   E0 01       CPX   #$01
 971B   D0 0E       BNE   $972B    ;nicht 1, dann weiter
 971D   AD 1B D0    LDA   VICMPR   ;Prioritaetsregister
 9720   A6 AA       LDX   $AA
 9722   1D 78 97    ORA   MBITTABS,X ; Bit setzen
 9725   8D 1B D0    STA   VICMPR
 9728   4C 36 97    JMP   $9736
```

```
 972B   AD 1B DO    LDA   VICMPR
 972E   A6 AA       LDX   $AA
 9730   3D 80 97    AND   MBITTABL,X ;Bit loeschen
 9733   8D 1B DO    STA   VICMPR

 9736   20 FC 81    JSR   SGETBYTC ;Art holen
 9739   E0 01       CPX   #$01     ;Multi-Colour-Sprite ?
 973B   D0 0E       BNE   $974B    ;nein, dann weiter
 973D   AD 1C DO    LDA   VICMMC   ;Multi-Colour-Registe
 9740   A6 AA       LDX   $AA
 9742   1D 78 97    ORA   MBITTABS,X ;Bit setzen
 9745   8D 1C DO    STA   VICMMC
 9748   4C 2A 82    JMP   ENDSMB   ;Befehl fertig

 974B   A6 AA       LDX   $AA
 974D   AD 1C DO    LDA   VICMMC
 9750   3D 80 97    AND   MBITTABL,X ;Bit loeschen
 9753   8D 1C DO    STA   VICMMC
 9756   4C 2A 82    JMP   ENDSMB   ;Befehl fertig

BEFMUSIC:
 9759   20 02 82    JSR   SGETBYTN ;Dauer holen
 975C   8E 96 CB    STX   MUSICDAU ;speichern
 975F   20 31 80    JSR   SCHKCOM  ;Komma ?
 9762   20 B7 86    JSR   SGETSTR  ;String holen
 9765   8E 9B CB    STX   MUSICADR  ;Stringadresse speichern
 9768   8D 9C CB    STA   MUSICADR+1
 976B   A0 00       LDY   #$00
 976D   8C 9D CB    STY   MUSICZ1  ;Zaehler = 0
 9770   A5 69       LDA   $69      ;Laenge des Strings
 9772   8D 9E CB    STA   MUSICLEN
 9775   4C 2A 82    JMP   ENDSMB   ;Befehl fertig

MBITTABS:
 9778   01 02 04 08 10 20 40 80  ;Tabelle zum Bit-Setzen

MBITTABL:
 9780   FE FD FB F7 EF DF BF 7F  ;Tabelle zum Bit-Loeschen

BITWTAB:
 9788   80 40 20 10 08 04 02 01  ;Tab. der Bit-Werigkeiten

BEFFLASH:
 9790   A9 0A       LDA   #$0A
 9792   8D C6 C5    STA   FLASHFL  ;Flag setzen
 9795   20 02 82    JSR   SGETBYTN ;Farbe holen
 9798   E0 10       CPX   #$10
 979A   B0 18       BCS   $97B4    ;groesser 16, dann fertig
 979C   A9 0A       LDA   #$0A
 979E   9D 04 C5    STA   FLASHFLS,X ;Flag fuer diese Farbe
 97A1   A0 00       LDY   #$00
```

```
 97A3   B1 7A      LDA  ($7A),Y   ;akt. Basic-Code
 97A5   C9 2C      CMP  #$2C      ;folgt Komma ?
 97A7   D0 0B      BNE  $97B4     ;nein, dann fertig
 97A9   20 02 82   JSR  SGETBYTN  ;speed holen
 97AC   8E C4 C5   STX  FLASHSP   ;speichern
 97AF   A9 00      LDA  #$00
 97B1   8D C5 C5   STA  FLASHZAE  ;Zaehler = 0
 97B4   4C 2A 82   JMP  ENDSMB    ;Befehl fertig

FNAT:
 97B7   20 02 82   JSR  SGETBYTN  ;holt Spalte
 97BA   8A         TXA            ;Spalte retten
 97BB   48         PHA
 97BC   20 FC 81   JSR  SGETBYTC  ;holt Zeile nach X
 97BF   18         CLC
 97C0   68         PLA
 97C1   A8         TAY            ;Spalte in Y
 97C2   20 F0 FF   JSR  $FFF0     ;Cursor setzen
 97C5   20 7F 80   JSR  SCHKKLZ   ;Klammer zu ?
 97C8   A9 00      LDA  #$00
 97CA   85 0D      STA  $0D       ;Stringflag loeschen
 97CC   20 79 00   JSR  CHRGOT    ;letzten Code holen
 97CF   4C 94 8C   JMP  $8C94     ;zur Auswertung eines
                                  ;arithmetischen Elements

SIRQ2:                       ;Fortsetzung der IRQ-Routine
 97D2   A5 FF      LDA  $FF  ;$FC bis $FF retten
 97D4   48         PHA
 97D5   A5 FE      LDA  $FE
 97D7   48         PHA
 97D8   A5 FD      LDA  $FD
 97DA   48         PHA
 97DB   A5 FC      LDA  $FC
 97DD   48         PHA
 97DE   AD C6 C5   LDA  FLASHFL ;FLASH aktiv ?
 97E1   C9 0A      CMP  #$0A
 97E3   F0 03      BEQ  $97E8   ;ja, dann weiter
 97E5   4C 50 98   JMP  SIRQ3   ;zur Fortsetzung

 97E8   EE C5 C5   INC  FLASHZAE ;Zaehler erhoehen
 97EB   AD C5 C5   LDA  FLASHZAE
 97EE   CD C4 C5   CMP  FLASHSP  ;gleich speed ?
 97F1   F0 03      BEQ  $97F6    ;ja, dann weiter
 97F3   4C 50 98   JMP  SIRQ3   ;zur Fortsetzung

 97F6   A9 D8      LDA  #$D8    ;$FE,$FF auf $D800
 97F8   85 FF      STA  $FF
 97FA   A9 00      LDA  #$00
 97FC   85 FE      STA  $FE
 97FE   AD 88 02   LDA  VIDRAMHI
 9801   85 FD      STA  $FD     ;$FC,$FD auf Video-RAM
```

```
 9803  A9 00      LDA  #$00
 9805  8D C5 C5   STA  FLASHZAE ;Zaehler = 0
 9808  85 FC      STA  $FC
 980A  A8         TAY
 980B  B1 FE      LDA  ($FE),Y  ;Zelle des Farb-RAM
 980D  29 0F      AND  #$0F
 980F  AA         TAX           ;Farb-Code nach X
 9810  BD 04 C5   LDA  FLASHFLS,X ;Flag fuer diese Farbe
 9813  F0 06      BEQ  $981B    ;nicht ges., dann weiter
 9815  B1 FC      LDA  ($FC),Y  ;Zelle aus Video-RAM
 9817  49 80      EOR  #$80     ;Bit 7 invert. (RVS)
 9819  91 FC      STA  ($FC),Y
 981B  E6 FE      INC  $FE      ;$FE,$FF erhoehen
 981D  D0 02      BNE  $9821
 981F  E6 FF      INC  $FF
 9821  E6 FC      INC  $FC      ;$FC,$FD erhoehen
 9823  D0 02      BNE  $9827
 9825  E6 FD      INC  $FD
 9827  A5 FF      LDA  $FF
 9829  C9 DB      CMP  #$DB     ;Endekriterium low
 982B  D0 DE      BNE  $980B    ;nein, dann Schleife
 982D  A5 FE      LDA  $FE
 982F  C9 E8      CMP  #$E8     ;Endekrtiterium high
 9831  D0 D8      BNE  $980B    ;nein, dann Schleife
 9833  4C 50 98   JMP  SIRQ3    ;zur IRQ-Fortsetzung

BEFPAGE:
 9836  20 02 82   JSR  SGETBYTN ;holt Wert
 9839  E0 00      CPX  #$00
 983B  D0 08      BNE  $9845    ;nicht null, dann weiter
 983D  A9 01      LDA  #$01
 983F  8D 8C C5   STA  PAGEFLAG ;Flag =1
 9842  4C 2A 82   JMP  ENDSMB   ;Befehl fertig

 9845  A9 0A      LDA  #$0A
 9847  8D 8C C5   STA  PAGEFLAG ;Flag = $0A (aktiv)
 984A  8E 8D C5   STX  PAGEWERT ;Wert speichern
 984D  4C 2A 82   JMP  ENDSMB   ;Befehl fertig

SIRQ3:
 9850  A5 D4      LDA  $D4      ;"-Flag bei Eingabe
 9852  F0 03      BEQ  $9857    ;nicht ges, dann weiter
 9854  4C 96 98   JMP  SIRQ4    ;zur IRQ-Fortsetzung

 9857  AC 8D 02   LDY  SHCOCTFL ;Shift/C=/CTRL-Flag
 985A  B9 92 98   LDA  SHWERTTAB,Y ;entspr. Wert
 985D  8D D6 C5   STA  INKEY    ;merken
 9860  A5 CB      LDA  $CB      ;gedrueckte Taste
 9862  C9 04      CMP  #$04     ;F1
 9864  F0 26      BEQ  $988C    ;ja, dann weiter
 9866  C9 05      CMP  #$05     ;F3
```

```
 9868  F0 1C       BEQ  $9886     ;ja ?
 986A  C9 06       CMP  #$06      ;F5
 986C  F0 12       BEQ  $9880     ;ja ?
 986E  C9 03       CMP  #$03      ;F7
 9870  F0 08       BEQ  $987A     ;ja ?
 9872  A9 00       LDA  #$00      ;keine Funktionstaste
 9874  8D D6 C5    STA  INKEY
 9877  4C 96 98    JMP  SIRQ4     ;zur IRQ-Fortsetzung

 987A  EE D6 C5    INC  INKEY     ;je nach Einsprungadresse
 987D  EE D6 C5    INC  INKEY     ;um 1 bis 7 erhoehen
 9880  EE D6 C5    INC  INKEY
 9883  EE D6 C5    INC  INKEY
 9886  EE D6 C5    INC  INKEY
 9889  EE D6 C5    INC  INKEY
 988C  EE D6 C5    INC  INKEY
 988F  4C 96 98    JMP  SIRQ4     ;zur IRQ-Fortsetzung

SHWERTTAB:
 9892  00 01 08 09             ;Tabelle fuer Shift und C=

SIRQ4:
 9896  A5 CC       LDA  $CC       ;blinkt Cursor ?
 9898  F0 03       BNE  $989D     ;ja, dann weiter
 989A  4C FF 98    JMP  SIRQ5     ;sonst zur IRQ-Fortsetzung

 989D  AD 46 C6    LDA  KEYFLAG   ;Key-Disable-Flag
 98A0  C9 0A       CMP  #$0A      ;gesetzt
 98A2  F0 F6       BEQ  $989A     ;nein, zur IRQ-Fortsetzung
 98A4  A5 D4       LDA  $D4       ;"-Flag
 98A6  D0 F2       BNE  $989A     ;gesetzt, dann zur IRQ-F.
 98A8  A5 CB       LDA  $CB       ;gedrueckte Taste
 98AA  CD 42 C6    CMP  LASTKEY   ;gleich letzter Taste ?
 98AD  F0 48       BEQ  $98F7     ;ja, dann weiter
 98AF  A5 D8       LDA  $D8       ;Insert-Modus ?
 98B1  D0 44       BNE  $98F7     ;ja, dann weiter
 98B3  AD D6 C5    LDA  INKEY     ;gedrueckte Funktionstaste
 98B6  F0 3F       BEQ  $98F7     ;keine, dann weiter
 98B8  AD D6 C5    LDA  INKEY
 98BB  85 FE       STA  $FE
 98BD  C9 11       CMP  #$11      ;Fkt.-Tastennr groesser 16
 98BF  B0 36       BCS  $98F7     ;ja, dann weiter
 98C1  A9 00       LDA  #$00
 98C3  85 FF       STA  $FF
 98C5  C6 FE       DEC  $FE
 98C7  06 FE       ASL  $FE
 98C9  26 FF       ROL  $FF
 98CB  06 FE       ASL  $FE
 98CD  26 FF       ROL  $FF
 98CF  06 FE       ASL  $FE
 98D1  26 FF       ROL  $FF
```

```
 98D3   06 FE       ASL  $FE
 98D5   26 FF       ROL  $FF        ;$FE,$FF = 16 * FTastennr.
 98D7   18          CLC
 98D8   A5 FE       LDA  $FE
 98DA   69 4D       ADC  #$4D
 98DC   85 FE       STA  $FE
 98DE   A5 FF       LDA  $FF
 98E0   69 C6       ADC  #$C6
 98E2   85 FF       STA  $FF        ;$FE,$FF = $FE,$FF + $C64D
 98E4   A0 00       LDY  #$00
 98E6   B1 FE       LDA  ($FE),Y    ;Zeichen aus KEYTAB
 98E8   F0 0B       BEQ  $98F5      ;0, dann weiter
 98EA   C0 0F       CPY  #$0F       ;15 Zeichen ?
 98EC   F0 07       BEQ  $98F5      ;ja, dann weiter
 98EE   99 77 02    STA  TASTPUFF,Y ;Zeichen in Tast.puff.
 98F1   C8          INY             ;naechstes Zeichen
 98F2   4C E6 98    JMP  $98E6      ;Schleife

 98F5   84 C6       STY  $C6        ;Anz. Zeich. im Tast.puff.
 98F7   A5 CB       LDA  $CB        ;gedrueckte Taste
 98F9   8D 42 C6    STA  LASTKEY    ;merken
 98FC   4C FF 98    JMP  SIRQ5      ;(unsinnig)

SIRQ5:
 98FF   AD 91 CB    LDA  PLAYFLAG   ;PLAY aktiv ?
 9902   C9 0A       CMP  #$0A
 9904   D0 03       BNE  $9909      ;nein, dann weiter
 9906   20 61 8A    JSR  SIRQPLAY
 9909   68          PLA             ;$FC bis $FF wiederherst.
 990A   85 FC       STA  $FC
 990C   68          PLA
 990D   85 FD       STA  $FD
 990F   68          PLA
 9910   85 FE       STA  $FE
 9912   68          PLA
 9913   85 FF       STA  $FF
 9915   4C 31 EA    JMP  BIRQ       ;zur normalen IRQ-Routine

BEFPLAY:
 9918   20 02 82    JSR  SGETBYTN   ;Wert (0,1,2) holen
 991B   E0 02       CPX  #$02
 991D   D0 0D       BNE  $992C      ;nicht 2, dann weiter
 991F   A9 00       LDA  #$00
 9921   8D 8A CB    STA  MUSICZ4    ;Zaehler = 0
 9924   A9 0A       LDA  #$0A
 9926   8D 91 CB    STA  PLAYFLAG   ;Flag setzen
 9929   4C 2A 82    JMP  ENDSMB     ;Befehl fertig

 992C   E0 01       CPX  #$01
 992E   D0 11       BNE  $9941      ;nicht 1, dann weiter
 9930   A9 00       LDA  #$00
```

```
 9932  8D 8A CB   STA  MUSICZ4  ;Zaehler = 0
 9935  A9 0A      LDA  #$0A
 9937  8D 91 CB   STA  PLAYFLAG ;Flag setzen
 993A  AD 91 CB   LDA  PLAYFLAG ;Flag testen
 993D  C9 0A      CMP  #$0A
 993F  F0 F9      BEQ  $993A    ;noch ges., dann Schleife
 9941  20 DE 8A   JSR  PLAYEND  ;Flag und Zaehler ruecks.
 9944  4C 2A 82   JMP  ENDSMB   ;Befehl fertig

BEFCENTRE:
 9947  20 DB 83   JSR  INCBASBZ ;Code ueberlesen
 994A  20 B7 86   JSR  SGETSTR  ;String holen
 994D  86 AC      STX  $AC      ;Stringadr. nach $AC,$AD
 994F  85 AD      STA  $AD
 9951  18         CLC
 9952  A9 28      LDA  #$28     ;40 Spalten
 9954  E5 69      SBC  $69      ;- Stringlaenge
 9956  85 6A      STA  $6A
 9958  46 6A      LSR  $6A      ;((40 - Len) div 2)
 995A  A9 1D      LDA  #$1D     ;Cursor right
 995C  20 D2 FF   JSR  BSOUT    ;ausgeben
 995F  C6 6A      DEC  $6A      ;Zaehler vermindern
 9961  A5 6A      LDA  $6A
 9963  C9 FF      CMP  #$FF     ;Zaehler abgelaufen ?
 9965  D0 F3      BNE  $995A    ;nein, dann Schleife
 9967  A0 00      LDY  #$00
 9969  4C 72 99   JMP  $9972    ;String ausgeben

 996C  B1 AC      LDA  ($AC),Y  ;Zeichen aus String
 996E  20 D2 FF   JSR  BSOUT    ;ausgeben
 9971  C8         INY           ;naechstes Zeichen
 9972  C4 69      CPY  $69      ;Stringlaenge erreicht ?
 9974  D0 F6      BNE  $996C    ;nein, dann Schleife
 9976  4C 2A 82   JMP  ENDSMB   ;Befehl fertig

BEFENVELOPE:
 9979  20 02 82   JSR  SGETBYTN ;Stimmenr. holen
 997C  20 6F 9A   JSR  X123     ;liegt X zw. 1 und 3 ?
 997F  CA         DEX
 9980  86 A6      STX  $A6      ;Stimmenr. - 1
 9982  20 31 80   JSR  SCHKCOM  ;Komma ?
 9985  A5 7A      LDA  $7A      ;Basic-Zeiger vermindern
 9987  D0 02      BNE  $998B
 9989  C6 7B      DEC  $7B
 998B  C6 7A      DEC  $7A
 998D  20 CC 99   JSR  GET2NYB  ;hole 2 Halbbytes (A,D)
 9990  A5 AA      LDA  $AA
 9992  85 A8      STA  $A8      ;AD in $A8 merken
 9994  20 31 80   JSR  SCHKCOM  ;Komma ?
 9997  A5 7A      LDA  $7A      ;Basic-Zeiger vermindern
 9999  D0 02      BNE  $999D
```

```
 999B   C6 7B       DEC   $7B
 999D   C6 7A       DEC   $7A
 999F   20 CC 99    JSR   GET2NYB   ;2 Halbbytes (S,R) holen
 99A2   06 A6       ASL   $A6       ;Stimmenr. verdoppeln
 99A4   A6 A6       LDX   $A6       ;als Index in Tab.
 99A6   BD 54 9A    LDA   SIDTAB,X  ;Registeradresse
 99A9   85 20       STA   $20       ;nach $20,$21
 99AB   E8          INX
 99AC   BD 54 9A    LDA   SIDTAB,X
 99AF   85 21       STA   $21
 99B1   46 A6       LSR   $A6       ;Stimmnr. wieder halbieren
 99B3   A0 06       LDY   #$06      ;Offset fuer Sust./Release
 99B5   A5 AA       LDA   $AA       ;SR-Wert
 99B7   91 20       STA   ($20),Y   ;setzen
 99B9   A4 A6       LDY   $A6
 99BB   99 A0 CB    STA   SRTAB,Y   ;in Tabelle ablegen
 99BE   A0 05       LDY   #$05      ;Offset fuer Attack/Decay
 99C0   A5 A8       LDA   $A8       ;AD-Wert
 99C2   91 20       STA   ($20),Y   ;setzen
 99C4   A4 A6       LDY   $A6
 99C6   99 A4 CB    STA   ADTAB,Y   ;in Tabelle ablegen
 99C9   4C 2A 82    JMP   ENDSMB    ;Befehl fertig

GET2NYB:                            ;hole 2 Werte von 0-15
 99CC   20 02 82    JSR   SGETBYTN  ;hole Byte-Wert
 99CF   20 6A 9A    JSR   XNYB      ;liegt X zw. 0 und 15 ?
 99D2   86 AA       STX   $AA
 99D4   06 AA       ASL   $AA
 99D6   06 AA       ASL   $AA
 99D8   06 AA       ASL   $AA
 99DA   06 AA       ASL   $AA       ;Wert = Wert * 16
 99DC   20 02 82    JSR   SGETBYTN  ;hole 2. Byte-Wert
 99DF   20 6A 9A    JSR   XNYB      ;liegt X zw. 0 und 15 ?
 99E2   8A          TXA
 99E3   05 AA       ORA   $AA       ;mit (16 * 1.Wert) odern
 99E5   85 AA       STA   $AA       ;in $AA ablegen
 99E7   60          RTS

BEFCGOTO:
 99E8   20 DB 83    JSR   INCBASBZ  ;Code ueberlesen
 99EB   20 0D 80    JSR   SFRMNUM   ;Wert holen
 99EE   20 16 80    JSR   SFACADR   ;in Adressformat umrechnen
 99F1   20 0F 81    JSR   SGOTO11   ;GOTO-Befehl durchfuehren
 99F4   4C 2A 82    JMP   ENDSMB    ;Befehl fertig

BEFWAVE:
 99F7   20 02 82    JSR   SGETBYTN  ;hole Stimmnr.
 99FA   20 6F 9A    JSR   X123      ;zwischen 1 und 3 ?
 99FD   CA          DEX
 99FE   86 AA       STX   $AA       ;Stimmnr. - 1 nach $AA
 9A00   20 31 80    JSR   SCHKCOM   ;Komma ?
```

```
 9A03   20 09 9A   JSR  BINCON     ;Binaeren Ausdruck holen
 9A06   4C 35 9A   JMP  $9A35      ;Fortsetzung s.u.

BINCON:
 9A09   A9 00      LDA  #$00
 9A0B   85 A6      STA  $A6        ;Bitzaehler
 9A0D   85 A8      STA  $A8        ;Ergebnis
 9A0F   A0 00      LDY  #$00
 9A11   B1 7A      LDA  ($7A),Y    ;akt. Basic-Zeichen
 9A13   C9 30      CMP  #$30       ;"0"
 9A15   F0 12      BEQ  $9A29      ;ja ?
 9A17   C9 31      CMP  #$31       ;"1"
 9A19   F0 05      BEQ  $9A20      ;ja ?
 9A1B   A9 04      LDA  #$04       ;Nr. f. 'not binary char'
 9A1D   4C 8C 88   JMP  SERROUT    ;Fehlermeldung ausgeben
 9A20   A4 A6      LDY  $A6
 9A22   A5 A8      LDA  $A8
 9A24   19 88 97   ORA  BITWTAB,Y  ;Ergebnis mit Wert odern
 9A27   85 A8      STA  $A8
 9A29   E6 A6      INC  $A6        ;Bitzaehler erhoehen
 9A2B   20 DB 83   JSR  INCBASBZ   ;naechstes Zeichen lesen
 9A2E   A5 A6      LDA  $A6
 9A30   C9 08      CMP  #$08       ;8 Zeichen gelesen ?
 9A32   D0 DB      BNE  $9A0F      ;nein, dann Schleife
 9A34   60         RTS

;Fortsetzung von BEFWAVE
 9A35   06 AA      ASL  $AA        ;Stimmnr. verdoppeln
 9A37   A4 AA      LDY  $AA        ;als Zeiger in SIDTAB
 9A39   B9 54 9A   LDA  SIDTAB,Y
 9A3C   85 20      STA  $20        ;$20,$21 auf SID-Reg.
 9A3E   C8         INY
 9A3F   B9 54 9A   LDA  SIDTAB,Y
 9A42   85 21      STA  $21
 9A44   A0 04      LDY  #$04       ;Offset fuer Waveform
 9A46   A5 A8      LDA  $A8        ;Wert fuer Waveform
 9A48   91 20      STA  ($20),Y    ;setzen
 9A4A   46 AA      LSR  $AA
 9A4C   A4 AA      LDY  $AA
 9A4E   99 4A C6   STA  WAVETAB,Y  ;in Tabelle ablegen
 9A51   4C 2A 82   JMP  ENDSMB     ;Befehl fertig

SIDTAB:
 9A54   00 D4 07 D4 0E D4          ;Tabelle der SID-Reg.-Adr.

DSCBMOUT:
 9A5A   A0 00      LDY  #$00
 9A5C   B9 A3 86   LDA  $86A3,Y    ;Zeich. a. Meld. 'ds-cbm'
 9A5F   F0 06      BEQ  $9A67      ;Ende ?
 9A61   20 D2 FF   JSR  BSOUT      ;Zeichen ausgeben
 9A64   C8         INY             ;naechstes Zeichen
```

```
 9A65  D0 F5       BNE  $9A5C      ;Schleife
 9A67  4C 74 91    JMP  BEFO       ;zum Leerbefehl

XNYB:
 9A6A  E0 10       CPX  #$10       ;X kleiner als 16 ?
 9A6C  B0 0A       BCS  $9A78      ;nein, dann Fehler
 9A6E  60          RTS

X123:
 9A6F  E0 04       CPX  #$04       ;X kleiner als 4 ?
 9A71  B0 05       BCS  $9A78      ;nein, dann Fehler
 9A73  E0 00       CPX  #$00       ;X=0 ?
 9A75  F0 01       BEQ  $9A78      ;ja, dann Fehler
 9A77  60          RTS

 9A78  A9 00       LDA  #$00       ;Nr. f. 'bad mode'
 9A7A  4C 8C 88    JMP  SERROUT    ;Fehler ausgeben

;Punkt im Multi-Colour-Modus
 9A7D  A5 F7       LDA  $F7        ;Punktfarbe
 9A7F  C9 0B       CMP  #$0B
 9A81  F0 4C       BEQ  $9ACF      ;=11, dann weiter
 9A83  C9 00       CMP  #$00
 9A85  D0 06       BNE  $9A8D      ;nicht 0, dann weiter
 9A87  20 B4 9A    JSR  GBITCLR    ;1.Bit loeschen
 9A8A  4C B4 9A    JMP  GBITCLR    ;2.Bit loeschen

 9A8D  C9 01       CMP  #$01
 9A8F  D0 06       BNE  $9A97      ;nicht 1, dann weiter
 9A91  20 B4 9A    JSR  GBITCLR    ;1.Bit loeschen
 9A94  4C AB 9A    JMP  GBITSET    ;2.Bit setzen

 9A97  C9 02       CMP  #$02
 9A99  D0 06       BNE  $9AA1      ;nicht 2, dann weiter
 9A9B  20 AB 9A    JSR  GBITSET    ;1.Bit setzen
 9A9E  4C B4 9A    JMP  GBITCLR    ;2.Bit loeschen

 9AA1  C9 03       CMP  #$03
 9AA3  D0 18       BNE  $9ABD      ;nicht 3, dann weiter
 9AA5  20 AB 9A    JSR  GBITSET    ;1.Bit setzen
 9AA8  4C AB 9A    JMP  GBITSET    ;2.Bit setzen

GBITSET:                           ;setzt Bit im Grafik-RAM
 9AAB  B1 A8       LDA  ($A8),Y      ;Zelle im Grafik-RAM
 9AAD  1D 30 93    ORA  GBITTABS,X ;mit Bit oderieren
 9AB0  91 A8       STA  ($A8),Y
 9AB2  E8          INX             ;naechstes Bit anpeilen
 9AB3  60          RTS

GBITCLR:                          ;loescht Bit im Grafik-RAM
 9AB4  B1 A8       LDA  ($A8),Y      ;Zelle im Grafik-RAM
```

```
 9AB6   3D 40 93   AND  GBITTABL,X ;entspr. Bit loeschen
 9AB9   91 A8      STA  ($A8),Y
 9ABB   E8         INX             ;naechstes Bit anpeilen
 9ABC   60         RTS

 9ABD   C9 04      CMP  #$04
 9ABF   D0 0E      BNE  $9ACF      ;nicht 4, dann weiter

;Punkt 'invertieren'
 9AC1   B1 A8      LDA  ($A8),Y
 9AC3   5D 30 93   EOR  GBITTABS,X ;1.Bit invertieren
 9AC6   91 A8      STA  ($A8),Y    ;(unsinnig)
 9AC8   E8         INX
 9AC9   5D 30 93   EOR  GBITTABS,X ;2.Bit invertieren
 9ACC   91 A8      STA  ($A8),Y
 9ACE   60         RTS

;Punkt testen
 9ACF   B1 A8      LDA  ($A8),Y
 9AD1   3D 30 93   AND  GBITTABS,X ;1.Bit ?
 9AD4   F0 04      BEQ  $9ADA     ;=0,dann weiter
 9AD6   A9 02      LDA  #$02
 9AD8   85 90      STA  $90       ;ST = 2
 9ADA   E8         INX            ;Zeiger auf 2.Bit
 9ADB   B1 A8      LDA  ($A8),Y
 9ADD   3D 30 93   AND  GBITTABS,X
 9AE0   F0 06      BEQ  $9AE8     ;=0,dann weiter
 9AE2   A5 90      LDA  $90
 9AE4   09 01      ORA  #$01      ;ST = ST or 1
 9AE6   85 90      STA  $90
 9AE8   60         RTS

 INCA6A7:                         ;$A6,$A7 erhoehen
 9AE9   E6 A6      INC  $A6
 9AEB   D0 02      BNE  $9AEF
 9AED   E6 A7      INC  $A7
 9AEF   60         RTS

 INCA8A9:                         ;$A8,$A9 erhoehen
 9AF0   E6 A8      INC  $A8
 9AF2   D0 02      BNE  $9AF6
 9AF4   E6 A9      INC  $A9
 9AF6   60         RTS

 BEFREPEAT:
 9AF7   AC 17 C6   LDY  SPREPEAT ;Stackpointer f. REPEAT
 9AFA   C0 09      CPY  #$09     ;kleiner als 9 ?
 9AFC   90 05      BCC  $9B03    ;ja, dann weiter

 9AFE   A9 09      LDA  #$09     ;Nr. F. 'stack too large'
 9B00   4C 8C 88   JMP  SERROUT  ;Fehler ausgeben
```

```
 9B03   C8          INY                 ;Pointer erhoehen
 9B04   A5 7A       LDA   $7A           ;Basic-Zeiger auf
 9B06   99 03 C6    STA   STACKREPEAT,Y ;Stapel legen
 9B09   C8          INY
 9B0A   A5 7B       LDA   $7B
 9B0C   99 03 C6    STA   STACKREPEAT,Y
 9B0F   8C 17 C6    STY   SPREPEAT ;Stapelzeiger speichern
 9B12   4C 74 91    JMP   BEFO          ;zum Leerbefehl

BEFUNTIL:
 9B15   20 DB 83    JSR   INCBASBZ ;Code ueberlesen
 9B18   AD 17 C6    LDA   SPREPEAT ;REPEAT-Stack
 9B1B   C9 02       CMP   #$02          ;groesser gleich 2 ?
 9B1D   B0 05       BCS   $9B24         ;ja, dann weiter

 9B1F   A9 08       LDA   #$08          ;Nr. f. 'until without r.'
 9B21   4C 8C 88    JMP   SERROUT       ;Fehler ausgeben

 9B24   20 3A 80    JSR   SFRMEVL       ;Ausdruck holen
 9B27   20 79 00    JSR   CHRGOT        ;letzten Basic-Code
 9B2A   A5 61       LDA   $61           ;Exponent von FAC
 9B2C   F0 09       BEQ   $9B37         ;null, dann weiter
 9B2E   CE 17 C6    DEC   SPREPEAT ;Stapelzeiger - 2
 9B31   CE 17 C6    DEC   SPREPEAT
 9B34   4C 2A 82    JMP   ENDSMB        ;Befehl fertig

 9B37   AC 17 C6    LDY   SPREPEAT ;Stapelzeiger f. REPEAT
 9B3A   B9 03 C6    LDA   STACKREPEAT,Y
 9B3D   85 7B       STA   $7B           ;Basic-Programmzaehle mit
 9B3F   88          DEY                 ;Inhalt vom Stapel laden
 9B40   B9 03 C6    LDA   STACKREPEAT,Y
 9B43   85 7A       STA   $7A
 9B45   4C 74 91    JMP   BEFO          ;zum Leerbefehl

BEFRETRACE:
 9B48   AE 01 C6    LDX   TRACEFLAG ;TRACE aktiv ?
 9B4B   E0 0A       CPX   #$0A
 9B4D   F0 05       BEQ   $9B54          ;ja, dann weiter
 9B4F   A9 00       LDA   #$00          ;Nr. f. 'bad mode'
 9B51   4C 8C 88    JMP   SERROUT       ;Fehler ausgeben

 9B54   20 E9 89    JSR   TRACESH1 ;Trace-Fenster malen
 9B57   4C 74 91    JMP   BEFO          ;zum Leerbefehl

BEFTRACE:
 9B5A   20 02 82    JSR   SGETBYTN   ;holt Wert (0 bzw. 10)
 9B5D   8E 01 C6    STX   TRACEFLAG ;als Flag speichern
 9B60   4C 2A 82    JMP   ENDSMB     ;Befehl fertig

BEFOPTION:
 9B63   20 02 82    JSR   SGETBYTN   ;holt Wert (0 bzw. 10)
```

```
 9B66   8E DC C5   STX   OPTFLAG  ;als Flag speichern
 9B69   4C 2A 82   JMP   ENDSMB   ;Befehl fertig

BEFIF:
 9B6C   20 DB 83   JSR   INCBASBZ ;Code ueberlesen
 9B6F   20 3A 80   JSR   SFRMEVL  ;Ausdruck auswerten
 9B72   20 79 00   JSR   CHRGOT   ;letztes Zeichen
 9B75   C9 89      CMP   #$89     ;Code fuer GOTO
 9B77   F0 05      BEQ   $9B7E    ;ja, dann weiter
 9B79   A9 A7      LDA   #$A7     ;Code fuer THEN
 9B7B   20 5E 80   JSR   SCHKZEI  ;folgt dieser Code ?
 9B7E   A5 61      LDA   $61      ;Exponent von Ergebnis
 9B80   8D CB C5   STA   IFFLAG   ;als Flag speichern
 9B83   D0 1F      BNE   $9BA4    ;wenn nicht 0, dann weiter
 9B85   A0 00      LDY   #$00
 9B87   B1 7A      LDA   ($7A),Y  ;akt. Basic-Code
 9B89   F0 16      BEQ   $9BA1    ;Zeilenende, dann fertig
 9B8B   C9 64      CMP   #$64     ;Simon's Praefix ?
 9B8D   D0 09      BNE   $9B98    ;nein, dann weiter
 9B8F   20 DB 83   JSR   INCBASBZ ;Praefix ueberlesen
 9B92   B1 7A      LDA   ($7A),Y  ;naechster Code
 9B94   C9 47      CMP   #$47     ;Code fuer ELSE
 9B96   F0 06      BEQ   $9B9E    ;ja, dann weiter
 9B98   20 DB 83   JSR   INCBASBZ ;naechsten Code
 9B9B   4C 87 9B   JMP   $9B87    ;Schleife

 9B9E   20 DB 83   JSR   INCBASBZ ;Code (ELSE) ueberlesen
 9BA1   4C 2A 82   JMP   ENDSMB   ;Befehl fertig

 9BA4   A0 00      LDY   #$00
 9BA6   B1 7A      LDA   ($7A),Y  ;akt. Code
 9BA8   C9 3A      CMP   #$3A     ;":"
 9BAA   B0 1F      BCS   $9BCB   ;groesser als 9,dann weiter
 9BAC   C9 20      CMP   #$20     ;" "
 9BAE   D0 06      BNE   $9BB6    ;nein ?
 9BB0   20 DB 83   JSR   INCBASBZ ;naechsten Code
 9BB3   4C A4 9B   JMP   $9BA4    ;Schleife

 9BB6   C9 60      CMP   #$60     ;Shift-Blank
 9BB8   F0 F6      BEQ   $9BB0    ;ja, dann s.o.
 9BBA   20 0D 80   JSR   SFRMNUM  ;Ausdruck holen
 9BBD   20 16 80   JSR   SFACADR  ;in Adressformat wandeln
 9BC0   20 55 80   JSR   SGOTO1   ;GOTO-Befehl ausfuehren
 9BC3   48         PHA
 9BC4   20 F3 82   JSR   BASROMEI ;Basic ein
 9BC7   68         PLA
 9BC8   4C AE A7   JMP   $A7AE    ;zur Interpreterschleife

 9BCB   A5 7A      LDA   $7A      ;Basic-Zeiger vermindern
 9BCD   D0 02      BNE   $9BD1
 9BCF   C6 7B      DEC   $7B
```

```
 9BD1  C6 7A        DEC   $7A
 9BD3  6C 08 03     JMP   ($0308) ;zur Ausfuehrung eines Bef.

BEFAUTO:
 9BD6  20 DB 83     JSR   INCBASBZ ;Code ueberlesen
 9BD9  20 0D 80     JSR   SFRMNUM  ;Wert holen
 9BDC  20 16 80     JSR   SFACADR  ;in Adressformat
 9BDF  8D FF 01     STA   AUTOZI+1 ;Anfangswert
 9BE2  8C FE 01     STY   AUTOZI
 9BE5  20 FC 81     JSR   SGETBYTC ;Inkrementwert holen
 9BE8  8E B1 C5     STX   AUTOINC
 9BEB  A9 0A        LDA   #$0A
 9BED  8D B2 C5     STA   AUTOFLAG ;Flag setzen
 9BF0  38           SEC
 9BF1  AD FE 01     LDA   AUTOZI   ;Anfangswert um Ink.Wert
 9BF4  ED B1 C5     SBC   AUTOINC  ;vermindern
 9BF7  8D FE 01     STA   AUTOZI
 9BFA  AD FF 01     LDA   AUTOZI+1 ;High-Byte
 9BFD  E9 00        SBC   #$00
 9BFF  8D FF 01     STA   AUTOZI+1
 9C02  4C 2A 82     JMP   ENDSMB   ;Befehl fertig

BEFRESET:
 9C05  20 DB 83     JSR   INCBASBZ ;Code ueberlesen
 9C08  20 0D 80     JSR   SFRMNUM  ;Wert holen
 9C0B  20 16 80     JSR   SFACADR  ;in Adressformat wandeln
 9C0E  85 15        STA   $15      ;nach $14,$15 und
 9C10  85 40        STA   $40      ;nach $3F,$40 (DATA-Zeile)
 9C12  84 14        STY   $14
 9C14  84 3F        STY   $3F
 9C16  20 70 80     JSR   SSTRPZ   ;Programmzeile suchen
 9C19  38           SEC
 9C1A  A5 5F        LDA   $5F
 9C1C  E9 01        SBC   #$01
 9C1E  A4 60        LDY   $60
 9C20  B0 01        BCS   $9C23
 9C22  88           DEY
 9C23  85 41        STA   $41      ;$41,$42 (DATA-Zeiger)
 9C25  84 42        STY   $42      ; = ($5F,$60) - 1
 9C27  4C 2A 82     JMP   ENDSMB   ;Befehl fertig

BEFCALL:
 9C2A  A9 31        LDA   #$31     ;Code fuer PROC
 9C2C  85 22        STA   $22      ;zu suchen
 9C2E  A9 0A        LDA   #$0A
 9C30  8D 97 C5     STA   PROCNFFL ;evtl. 'proc not found'
 9C33  20 39 9C     JSR   SUCHCODE ;Code (in $22) suchen
 9C36  4C 66 9C     JMP   $9C66

SUCHCODE:
 9C39  18           CLC
```

```
 9C3A  A5 2B        LDA  $2B        ;$20,$21 = 4 + $2B,$2C
 9C3C  69 04        ADC  #$04                (Basic-Start)
 9C3E  85 20        STA  $20
 9C40  A5 2C        LDA  $2C
 9C42  69 00        ADC  #$00
 9C44  85 21        STA  $21

SUCHCODE1:
 9C46  A0 00        LDY  #$00
 9C48  B1 20        LDA  ($20),Y    ;Basic-Code
 9C4A  C9 64        CMP  #$64       ;Simon's Praefix ?
 9C4C  D0 0A        BNE  $9C58      ;nein, dann weiter
 9C4E  20 DC 9C     JSR  INC$20     ;Zeiger erhoehen
 9C51  B1 20        LDA  ($20),Y
 9C53  C5 22        CMP  $22        ;zu suchendes Zeichen
 9C55  D0 01        BNE  $9C58      ;nicht gef., dann weiter
 9C57  60           RTS             ;sonst fertig

 9C58  B1 20        LDA  ($20),Y
 9C5A  F0 27        BEQ  $9C83      ;Zeilenende, dann weiter
 9C5C  C9 22        CMP  #$22       ;Anfuehrungszeichen
 9C5E  F0 6C        BEQ  $9CCC      ;ja, dann weiter

SUCHCODEN:
 9C60  20 DC 9C     JSR  INC$20      ;Zeiger erhoehen
 9C63  4C 46 9C     JMP  SUCHCODE1 ;zur Suchschleife

;Fortsetzung von BEFCALL
 9C66  C8           INY
 9C67  B1 7A        LDA  ($7A),Y    ;akt. Code
 9C69  F0 0B        BEQ  $9C76      ;Zeilenende, dann weiter
 9C6B  D1 20        CMP  ($20),Y    ;mit zu such. Namen vgl.
 9C6D  D0 48        BNE  $9CB7      ;ungleich ?
 9C6F  B1 20        LDA  ($20),Y
 9C71  F0 44        BEQ  $9CB7      ;Name zu Ende ?
 9C73  4C 66 9C     JMP  $9C66      ;Schleife

 9C76  A0 00        LDY  #$00
 9C78  A5 20        LDA  $20        ;Basic-Zeiger ($7A,$7B)
 9C7A  85 7A        STA  $7A        ;mit $20,$21 laden
 9C7C  A5 21        LDA  $21
 9C7E  85 7B        STA  $7B
 9C80  4C 30 9F     JMP  BEFPROC    ;zum PROC-Befehl

;Fortsetzung von SUCHCODE (bzw. SUCHCODEN)
 9C83  20 DC 9C     JSR  INC$20     ;Zeiger erhoehen
 9C86  B1 20        LDA  ($20),Y    ;Vorwaertszeiger low
 9C88  D0 05        BNE  $9C8F      ;ungleich 0 ?
 9C8A  A9 01        LDA  #$01
 9C8C  8D DD C5     STA  PENDFLAG ;Flag f. Prog.end setzen
 9C8F  20 DC 9C     JSR  INC$20     ;Zeiger erhoehen
```

```
 9C92  B1 20      LDA  ($20),Y   ;Vorwaertszeiger high
 9C94  D0 2A      BNE  $9CC0     ;ungleich 0 ?
 9C96  AD DD C5   LDA  PENDFLAG
 9C99  C9 01      CMP  #$01
 9C9B  D0 23      BNE  $9CC0     ;Prog. n. zu Ende, dann w.
 9C9D  AD 97 C5   LDA  PROCNFFL  ;Fehlermeldung ausgeben ?
 9CA0  C9 0A      CMP  #$0A
 9CA2  F0 05      BEQ  $9CA9     ;ja ?
 9CA4  68         PLA            ;Stack bereinigen
 9CA5  68         PLA
 9CA6  4C 2A 82   JMP  ENDSMB    ;Befehl beenden

 9CA9  A9 01      LDA  #$01
 9CAB  8D 97 C5   STA  PROCNFFL  ;Flag ruecksetzen
 9CAE  CE 2C C6   DEC  SPEXEC    ;EXEC-Stack bereinigen
 9CB1  CE 2C C6   DEC  SPEXEC
 9CB4  4C 8C 88   JMP  SERROUT   ;Fehler ausgeben

;Fortsetzung von BEFCALL
 9CB7  20 DC 9C   JSR  INC$20     ;Zeiger erhoehen
 9CBA  20 46 9C   JSR  SUCHCODE1  ;Code weitersuchen
 9CBD  4C 66 9C   JMP  $9C66      ;s.o.

;Fortsetzung von SUCHCODE
 9CC0  20 DC 9C   JSR  INC$20
 9CC3  20 DC 9C   JSR  INC$20
 9CC6  20 DC 9C   JSR  INC$20
 9CC9  4C 46 9C   JMP  SUCHCODE1

;Fortsetzung von SUCHCODE (bei Anfuehrungszeichen)
 9CCC  20 DC 9C   JSR  INC$20    ;Zeiger weiterstellen
 9CCF  A0 00      LDY  #$00
 9CD1  B1 20      LDA  ($20),Y
 9CD3  F0 AE      BEQ  $9C83     ;Zeilenende ?
 9CD5  C9 22      CMP  #$22      ;2. Anfuehrungszeichen
 9CD7  F0 ED      BEQ  $9CC6     ;ja, dann nach oben
 9CD9  4C CC 9C   JMP  $9CCC     ;Schleife

INC$20:                          ;$20,$21 erhoehen
 9CDC  E6 20      INC  $20
 9CDE  D0 02      BNE  $9CE2
 9CE0  E6 21      INC  $21
 9CE2  60         RTS

BEFEXEC:
 9CE3  A5 7A      LDA  $7A       ;Prg.-Zaehler sichern
 9CE5  48         PHA
 9CE6  A5 7B      LDA  $7B
 9CE8  48         PHA
 9CE9  A0 00      LDY  #$00
 9CEB  B1 7A      LDA  ($7A),Y   ;akt. Zeichen
```

```
 9CED   F0 06      BEQ  $9CF5     ;Zeilenende ?
 9CEF   20 DB 83   JSR  INCBASBZ  ;Basic-Zeiger eroehen
 9CF2   4C E9 9C   JMP  $9CE9     ;Schleife

 9CF5   AC 2C C6   LDY  SPEXEC    ;EXEC-Stapel-Zeiger
 9CF8   C0 09      CPY  #$09
 9CFA   90 05      BCC  $9D01     ;Stapel nicht voll ?
 9CFC   A9 09      LDA  #$09      ;Nr. von 'stack too large'
 9CFE   4C 8C 88   JMP  SERROUT   ;Fehler ausgeben

 9D01   C8         INY            ;Zeiger erhoehen
 9D02   A5 7A      LDA  $7A       ;akt. Prg.z. auf Stapel
 9D04   99 18 C6   STA  STACKEXEC,Y
 9D07   C8         INY
 9D08   A5 7B      LDA  $7B
 9D0A   99 18 C6   STA  STACKEXEC,Y
 9D0D   8C 2C C6   STY  SPEXEC    ;Stapelzeiger speichern
 9D10   68         PLA            ;alten Programmzeiger
 9D11   85 7B      STA  $7B       ;wiederherstellen
 9D13   68         PLA
 9D14   85 7A      STA  $7A
 9D16   4C 2A 9C   JMP  BEFCALL   ;zum CALL-Befehl

BEFENDPROC:
 9D19   AC 2C C6   LDY  SPEXEC
 9D1C   C0 02      CPY  #$02
 9D1E   B0 05      BCS  $9D25     ;EXEC-Stapel nicht leer ?
 9D20   A9 06      LDA  #$06      ;Nr. f. 'end proc w.o. e'
 9D22   4C 8C 88   JMP  SERROUT   ;Fehler ausgeben

 9D25   B9 18 C6   LDA  STACKEXEC,Y
 9D28   85 7B      STA  $7B       ;Programmzaehler mit
 9D2A   88         DEY            ;Inhalt von Stapel laden
 9D2B   B9 18 C6   LDA  STACKEXEC,Y
 9D2E   85 7A      STA  $7A
 9D30   88         DEY
 9D31   8C 2C C6   STY  SPEXEC    ;Stapelzeiger speichern
 9D34   4C 2A 82   JMP  ENDSMB    ;Befehl fertig

BEFEXIT:
 9D37   20 DB 83   JSR  INCBASBZ  ;Code ueberlesen
 9D3A   20 DB 83   JSR  INCBASBZ  ;Blank ueberlesen
 9D3D   A9 8B      LDA  #$8B      ;Code fuer IF
 9D3F   20 5E 80   JSR  SCHKZEI   ;folgt dieser Code ?
 9D42   20 3A 80   JSR  SFRMEVL   ;Ausdruck holen
 9D45   20 79 00   JSR  CHRGOT    ;(unsinnig)
 9D48   A5 61      LDA  $61       ;Exponent von Ergebnis
 9D4A   D0 03      BNE  $9D4F     ;nicht null, dann weiter
 9D4C   4C 2A 82   JMP  ENDSMB    ;sonst nur Befehl beenden

 9D4F   A9 36      LDA  #$36      ;Code fuer END LOOP
```

```
 9D51   85 22       STA  $22        ;als Suchzeichen merken
 9D53   A5 7A       LDA  $7A        ;$20,$21 = $7A,$7B
 9D55   85 20       STA  $20        ;(ab Prg.z. suchen)
 9D57   A5 7B       LDA  $7B
 9D59   85 21       STA  $21
 9D5B   20 46 9C    JSR  SUCHCODE1 ;Code ab hier suchen
 9D5E   A5 20       LDA  $20        ;Prg.z. mit gefundener
 9D60   85 7A       STA  $7A        ;Adresse laden
 9D62   A5 21       LDA  $21
 9D64   85 7B       STA  $7B
 9D66   CE 41 C6    DEC  SPLOOP     ;Stapel f. LOOP bereinigen
 9D69   CE 41 C6    DEC  SPLOOP
 9D6C   4C 74 91    JMP  BEFO       ;zum Leerbefehl

BEFENDLOOP:
 9D6F   AC 41 C6    LDY  SPLOOP
 9D72   C0 02       CPY  #$02
 9D74   B0 05       BCS  $9D7B      ;LOOP-Stapel nicht leer ?
 9D76   A9 07       LDA  #$07       ;Nr. f. 'end loop w.o. l.'
 9D78   4C 8C 88    JMP  SERROUT    ;Fehler ausgeben

 9D7B   B9 2D C6    LDA  STACKLOOP,Y
 9D7E   85 7B       STA  $7B        ;Basic-Prg.Z. mit Wert aus
 9D80   88          DEY             ;LOOP-Stapel laden
 9D81   B9 2D C6    LDA  STACKLOOP,Y
 9D84   85 7A       STA  $7A
 9D86   4C 74 91    JMP  BEFO       ;zum Leerbefehl

BEFONKEY:
 9D89   A9 00       LDA  #$00
 9D8B   8D EA C5    STA  ONKEYFLAG ;Flag loeschen
 9D8E   20 DB 83    JSR  INCBASBZ ;Code ueberlesen
 9D91   A5 7A       LDA  $7A        ;Prg.z. als Onkey-Zeiger
 9D93   8D E8 C5    STA  ONKEYZEIG
 9D96   A5 7B       LDA  $7B
 9D98   8D E9 C5    STA  ONKEYZEIG+1
 9D9B   20 B7 86    JSR  SGETSTR ;holt String (ohne Wirkung)
 9D9E   A9 0A       LDA  #$0A
 9DA0   8D EA C5    STA  ONKEYFLAG ;Flag setzen
 9DA3   A0 00       LDY  #$00
 9DA5   B1 7A       LDA  ($7A),Y    ;akt. Basic-Code
 9DA7   F0 06       BEQ  $9DAF      ;Zeilenende ?
 9DA9   20 DB 83    JSR  INCBASBZ   ;naechstes Zeichen
 9DAC   4C A3 9D    JMP  $9DA3      ;Schleife

 9DAF   4C 2A 82    JMP  ENDSMB     ;Befehl fertig

BEFDISABLE:
 9DB2   A9 00       LDA  #$00
 9DB4   8D EA C5    STA  ONKEYFLAG ;Flag loeschen
 9DB7   4C 74 91    JMP  BEFO       ;zum Leerbefehl
```

```
ONKEY1:
 9DBA  A5 7A     LDA  $7A        ;Programmzaehler retten
 9DBC  48        PHA
 9DBD  A5 7B     LDA  $7B
 9DBF  48        PHA
 9DC0  AD E8 C5  LDA  ONKEYZEIG  ;Programmz. mit Onkey-
 9DC3  85 7A     STA  $7A        ;Zeiger laden
 9DC5  AD E9 C5  LDA  ONKEYZEIG+1
 9DC8  85 7B     STA  $7B
 9DCA  A9 00     LDA  #$00
 9DCC  8D EA C5  STA  ONKEYFLAG  ;Flag ruecksetzen
 9DCF  20 B7 86  JSR  SGETSTR    ;String holen
 9DD2  85 AD     STA  $AD        ;Adresse nach $AC,$AD
 9DD4  86 AC     STX  $AC
 9DD6  20 67 80  JSR  SCHKCOM1   ;Komma ?
 9DD9  20 E4 FF  JSR  GETIN      ;Zeich. von Tastatur holen
 9DDC  A0 0A     LDY  #$0A
 9DDE  8C EA C5  STY  ONKEYFLAG  ;Flag wieder setzen
 9DE1  8D EC C5  STA  KEYON      ;Zeichen speichern
 9DE4  A0 00     LDY  #$00
 9DE6  B1 AC     LDA  ($AC),Y    ;String durchsuchen
 9DE8  CD EC C5  CMP  KEYON      ;nach gedr. Taste
 9DEB  F0 10     BEQ  $9DFD      ;gefunden ?
 9DED  C9 5D     CMP  #$5D       ;Eckige Klammer zu ?
 9DEF  F0 0C     BEQ  $9DFD      ;ja ?
 9DF1  C8        INY             ;naechstes Zeichen
 9DF2  C4 69     CPY  $69        ;String zu Ende ?
 9DF4  D0 F0     BNE  $9DE6      ;nein, dann Schleife
 9DF6  68        PLA             ;alten Prg.Z. restaurieren
 9DF7  85 7B     STA  $7B
 9DF9  68        PLA
 9DFA  85 7A     STA  $7A
 9DFC  60        RTS             ;fertig

 9DFD  AD EC C5  LDA  KEYON      ;gedruecktes Zeichen
 9E00  85 90     STA  $90        ;in Statusvariable ST
 9E02  68        PLA
 9E03  8D F0 C5  STA  BASBZK     ;alten Prg.z. in BASBZK
 9E06  68        PLA             ;und BASBZK+1 sichern
 9E07  8D F1 C5  STA  BASBZK+1
 9E0A  68        PLA             ;Stack bereinigen
 9E0B  68        PLA
 9E0C  4C 2A 82  JMP  ENDSMB     ;Befehl beenden

BEFRESUME:
 9E0F  AD F0 C5  LDA  BASBZK     ;gesicherten Wert
 9E12  85 7B     STA  $7B        ;in Prg.Z. uebertragen
 9E14  AD F1 C5  LDA  BASBZK+1
 9E17  85 7A     STA  $7A
 9E19  20 F3 82  JSR  BASROMEI   ;Basic ein
 9E1C  A9 0A     LDA  #$0A
```

```
 9E1E   8D EA C5   STA  ONKEYFLAG ;Flag loeschen
 9E21   4C A6 91   JMP  $91A6

BEFLOOP:
 9E24   AC 41 C6   LDY  SPLOOP
 9E27   C0 09      CPY  #$09
 9E29   90 05      BCC  $9E30    ;LOOP-Stack nicht voll ?
 9E2B   A9 09      LDA  #$09     ;Nr. f. 'stack too large'
 9E2D   4C 8C 88   JMP  SERROUT  ;Fehler ausgeben

 9E30   C8         INY           ;Zeiger erhoehen
 9E31   A5 7A      LDA  $7A      ;Prg.Z. auf Stapel legen
 9E33   99 2D C6   STA  STACKLOOP,Y
 9E36   C8         INY
 9E37   A5 7B      LDA  $7B
 9E39   99 2D C6   STA  STACKLOOP,Y
 9E3C   8C 41 C6   STY  SPLOOP   ;Stapelzeiger speichern
 9E3F   4C 74 91   JMP  BEFO     ;zum Leerbef.

BEFDELAY:
 9E42   20 02 82   JSR  SGETBYTN ;Wert holen
 9E45   8E F8 C5   STX  DELAY    ;speichern
 9E48   4C 2A 82   JMP  ENDSMB   ;fertig

BEFSECURE:
 9E4B   20 02 82   JSR  SGETBYTN ;Wert ueberlesen
 9E4E   A9 41      LDA  #$41     ;Code fuer DISAPA
 9E50   85 22      STA  $22      ;als Suchzeichen
 9E52   8D 97 C5   STA  PROCNFFL ;kein Fehler, wenn n. gef.
 9E55   20 39 9C   JSR  SUCHCODE ;Code suchen
 9E58   A5 20      LDA  $20      ;$20,$21 vermindern
 9E5A   D0 02      BNE  $9E5E
 9E5C   C6 21      DEC  $21
 9E5E   C6 20      DEC  $20
 9E60   A9 00      LDA  #$00     ;Zeilenendemarkierung
 9E62   91 20      STA  ($20),Y  ;hier einfuegen
 9E64   20 60 9C   JSR  SUCHCODEN ;Code nochmal suchen
 9E67   A9 3A      LDA  #$3A     ;":"
 9E69   91 20      STA  ($20),Y  ;dort einsetzen
 9E6B   4C 58 9E   JMP  $9E58    ;Schleife (bis Prg.Ende)

BEFONERR:
 9E6E   A9 00      LDA  #$00
 9E70   8D FB C5   STA  ONERRORFLAG ;Flag loeschen
 9E73   20 DB 83   JSR  INCBASBZ    ;Code ueberlesen
 9E76   A5 7A      LDA  $7A        ;Prg.Z in ONERRZEIG und
 9E78   8D F9 C5   STA  ONERRZEIG  ;ONERRZEIG+1 speichern
 9E7B   A5 7B      LDA  $7B
 9E7D   8D FA C5   STA  ONERRZEIG+1
 9E80   A9 0A      LDA  #$0A
 9E82   8D FB C5   STA  ONERRORFLAG ;Flag setzen
```

```
 9E85   A0 00      LDY  #$00
 9E87   B1 7A      LDA  ($7A),Y   ;akt. Code
 9E89   F0 06      BEQ  $9E91     ;Zeilenende ?
 9E8B   20 DB 83   JSR  INCBASBZ  ;naechsten Code
 9E8E   4C 85 9E   JMP  $9E85     ;Schleife

 9E91   4C 2A 82   JMP  ENDSMB    ;fertig

BEFNOERR:
 9E94   A9 00      LDA  #$00
 9E96   8D FB C5   STA  ONERRORFLAG ;Flag loeschen
 9E99   4C 74 91   JMP  BEF0      ;zum Leerbefehl

SWARM:
 9E9C   8E FC C5   STX  ERRN      ;Fehlernummer speichern
 9E9F   20 F3 82   JSR  BASROMEI ;Basic ein
 9EA2   AD FB C5   LDA  ONERRORFLAG ;ON ERROR aktiv ?
 9EA5   C9 0A      CMP  #$0A
 9EA7   F0 06      BEQ  $9EAF     ;ja, dann weiter
 9EA9   AE FC C5   LDX  ERRN      ;Fehlernummer holen
 9EAC   4C 8B E3   JMP  $E38B     ;zur Std.-Fehlerroutine

 9EAF   AD FA C5   LDA  ONERRZEIG+1 ;Prg.Z. mit Wert von
 9EB2   85 7B      STA  $7B       ;ONERRROR-Zeiger laden
 9EB4   AD F9 C5   LDA  ONERRZEIG
 9EB7   85 7A      STA  $7A
 9EB9   A5 39      LDA  $39       ;akt. Zeilennr. in ERRLN
 9EBB   8D 1D C5   STA  ERRLN     ;und ERRLN+1 ablegen
 9EBE   A5 3A      LDA  $3A
 9EC0   8D 1E C5   STA  ERRLN+1
 9EC3   A9 00      LDA  #$00
 9EC5   8D 17 C6   STA  SPREPEAT ;Stacks initialisieren
 9EC8   8D 2C C6   STA  SPEXEC
 9ECB   8D 41 C6   STA  SPLOOP
 9ECE   4C 2A 82   JMP  ENDSMB    ;fertig

BEFOUT:
 9ED1   20 F3 82   JSR  BASROMEI ;Basic ein
 9ED4   AE FC C5   LDX  ERRN      ;Fehlernummer holen
 9ED7   4C 8B E3   JMP  $E38B     ;zur Std.-Fehlerroutine

BEFOLD:
 9EDA   A9 04      LDA  #$04
 9EDC   85 20      STA  $20       ;$20,$21 auf Basic-Anf. +3
 9EDE   A5 2C      LDA  $2C
 9EE0   85 21      STA  $21
 9EE2   A0 00      LDY  #$00
 9EE4   20 DC 9C   JSR  INC$20    ;Zeiger erhoehen
 9EE7   B1 20      LDA  ($20),Y
 9EE9   D0 F9      BNE  $9EE4     ;kein Zei.ende, dann Schl.
 9EEB   20 DC 9C   JSR  INC$20    ;Zeiger erhoehen
```

```
 9EEE   A5 20       LDA  $20       ;Vorwaertsz. restaurieren
 9EF0   91 2B       STA  ($2B),Y
 9EF2   C8          INY
 9EF3   A5 21       LDA  $21
 9EF5   91 2B       STA  ($2B),Y
 9EF7   88          DEY
 9EF8   A9 01       LDA  #$01
 9EFA   85 2D       STA  $2D
 9EFC   A2 03       LDX  #$03     ;3 Nullen werden gesucht
 9EFE   20 DC 9C    JSR  INC$20   ;naechstes Zeichen
 9F01   B1 20       LDA  ($20),Y
 9F03   D0 F7       BNE  $9EFC    ;innere Schleife
 9F05   CA          DEX           ;Zaehler vermindern
 9F06   D0 F6       BNE  $9EFE    ;aeussere Schleife
 9F08   20 DC 9C    JSR  INC$20   ;Zeiger erhoehen
 9F0B   A5 20       LDA  $20      ;Zeiger auf Ende der
 9F0D   85 31       STA  $31      ;Arrays ($31,$32) auf mom.
 9F0F   A5 21       LDA  $21      ;Zeiger
 9F11   85 32       STA  $32
 9F13   A2 03       LDX  #$03
 9F15   B5 2F       LDA  $2F,X    ;Zeiger uebertragen
 9F17   95 2D       STA  $2D,X
 9F19   CA          DEX           ;Schleife fuer 4 Byte
 9F1A   10 F9       BPL  $9F15    ;noch nicht fertig ?
 9F1C   4C 74 91    JMP  BEF0     ;zum Leerbefehl

BEFRCOMP:
 9F1F   20 DB 83    JSR  INCBASBZ ;Code ueberlesen
 9F22   20 F3 82    JSR  BASROMEI ;Basic ein
 9F25   20 DB 83    JSR  INCBASBZ
 9F28   AD CB C5    LDA  IFFLAG    ;altes IF-Flag
 9F2B   85 61       STA  $61       ;nach $61 schreiben
 9F2D   4C 7E 9B    JMP  $9B7E     ;in den IF-Befehl

BEFPROC:
 9F30   A0 00       LDY  #$00
 9F32   B1 7A       LDA  ($7A),Y
 9F34   F0 06       BEQ  $9F3C     ;Zeilenende ?
 9F36   20 DB 83    JSR  INCBASBZ
 9F39   4C 32 9F    JMP  $9F32     ;Schleife

 9F3C   4C 2A 82    JMP  ENDSMB    ;fertig

BEFDUMP:
 9F3F   A5 2D       LDA  $2D      ;Start der Var. nach
 9F41   85 20       STA  $20      ;$20,$21 kopieren
 9F43   A5 2E       LDA  $2E
 9F45   85 21       STA  $21
 9F47   A9 0D       LDA  #$0D     ;(CR)
 9F49   20 D2 FF    JSR  BSOUT    ;ausgeben
 9F4C   A5 21       LDA  $21      ;Zeiger high
```

```
 9F4E  A0 00      LDY  #$00
 9F50  C5 30      CMP  $30       ;mit Start d. Array vgl.
 9F52  90 0D      BCC  $9F61     ;kleiner, dann weiter
 9F54  F0 02      BEQ  $9F58     ;gleich, dann low-B. vgl.
 9F56  B0 06      BCS  $9F5E     ;groesser, dann Ende
 9F58  A5 20      LDA  $20
 9F5A  C5 2F      CMP  $2F
 9F5C  90 03      BCC  $9F61
 9F5E  4C 74 91   JMP  BEFO      ;zum Leerbefehl; Ende

 9F61  B1 20      LDA  ($20),Y   ;1.Zeichen Variablenname
 9F63  85 A6      STA  $A6
 9F65  09 80      ORA  #$80
 9F67  C5 A6      CMP  $A6       ;war MSB gesetzt ?
 9F69  D0 40      BNE  $9FAB     ;nein, dann weiter
 9F6B  29 7F      AND  #$7F
 9F6D  20 D2 FF   JSR  BSOUT     ;Zeichen ausgeben
 9F70  20 39 A0   JSR  INC$20    ;naechstes Zeichen
 9F73  B1 20      LDA  ($20),Y
 9F75  29 7F      AND  #$7F
 9F77  20 D2 FF   JSR  BSOUT     ;ausgeben
 9F7A  A9 25      LDA  #$25      ;"%"
 9F7C  20 D2 FF   JSR  BSOUT     ;ausgeben
 9F7F  20 40 A0   JSR  OUT=      ;"=" ausgeben
 9F82  20 39 A0   JSR  INC$20    ;Zeiger erhoehen
 9F85  B1 20      LDA  ($20),Y   ;High-Byte
 9F87  85 62      STA  $62
 9F89  20 39 A0   JSR  INC$20
 9F8C  B1 20      LDA  ($20),Y   ;Low-Byte
 9F8E  85 63      STA  $63
 9F90  A2 90      LDX  #$90
 9F92  38         SEC
 9F93  20 88 80   JSR  SXFLP     ;nach Fliessk. wandeln
 9F96  20 B5 80   JSR  SFACASC   ;in Text umwandeln
 9F99  20 2B A0   JSR  OUTASC    ;Text ausgeben (Wert)
 9F9C  20 39 A0   JSR  INC$20    ;3 Byte ueberlesen
 9F9F  20 39 A0   JSR  INC$20
 9FA2  20 39 A0   JSR  INC$20
 9FA5  20 39 A0   JSR  INC$20
 9FA8  4C 47 9F   JMP  $9F47     ;zur naechsten Variablen

;hierher, wenn keine Integer-Variable
 9FAB  29 7F      AND  #$7F
 9FAD  20 D2 FF   JSR  BSOUT     ;1. Zeichen ausgeben
 9FB0  20 39 A0   JSR  INC$20    ;Zeiger auf 2. Zeichen
 9FB3  B1 20      LDA  ($20),Y
 9FB5  85 A6      STA  $A6
 9FB7  09 80      ORA  #$80
 9FB9  C5 A6      CMP  $A6       ;war MSB gesetzt ?
 9FBB  D0 44      BNE  $A001     ;nein, dann weiter
 9FBD  29 7F      AND  #$7F
```

```
 9FBF   20 D2 FF    JSR   BSOUT      ;2. Zeichen ausgeben
 9FC2   A9 24       LDA   #$24       ;"$"
 9FC4   20 D2 FF    JSR   BSOUT      ;ausgeben
 9FC7   20 40 A0    JSR   OUT=
 9FCA   A9 22       LDA   #$22       ;Anf.zeichen
 9FCC   20 D2 FF    JSR   BSOUT      ;ausgeben
 9FCF   20 39 A0    JSR   INC$20
 9FD2   B1 20       LDA   ($20),Y    ;Stringlaenge
 9FD4   85 A8       STA   $A8
 9FD6   20 39 A0    JSR   INC$20
 9FD9   B1 20       LDA   ($20),Y    ;Stringadr. low
 9FDB   85 23       STA   $23
 9FDD   20 39 A0    JSR   INC$20
 9FE0   B1 20       LDA   ($20),Y    ;Stringadr. high
 9FE2   85 24       STA   $24
 9FE4   A0 00       LDY   #$00
 9FE6   B1 23       LDA   ($23),Y    ;Zeichen des Strings
 9FE8   20 D2 FF    JSR   BSOUT
 9FEB   C8          INY
 9FEC   C4 A8       CPY   $A8        ;Stringlaenge erreicht ?
 9FEE   D0 F6       BNE   $9FE6      ;nein, dann Schleife
 9FF0   A9 22       LDA   #$22       ;Anfuehrungszeichen
 9FF2   20 D2 FF    JSR   BSOUT      ;ausgeben
 9FF5   20 39 A0    JSR   INC$20     ;2 Byte ueberlesen
 9FF8   20 39 A0    JSR   INC$20
 9FFB   20 39 A0    JSR   INC$20
 9FFE   4C 47 9F    JMP   $9F47      ;zur naechsten Variablen

;hierher, wenn Fliesskomma-Variable
 A001   29 7F       AND   #$7F
 A003   20 D2 FF    JSR   BSOUT      ;2. Zeichen ausgeben
 A006   20 39 A0    JSR   INC$20
 A009   A5 20       LDA   $20        ;Konstantenadresse
 A00B   A4 21       LDY   $21        ;ist akt. Zaehlerstand
 A00D   20 A3 80    JSR   SFLDFACK   ;Konstante in FAC bringen
 A010   20 39 A0    JSR   INC$20     ;insges. 5 Byte ueberlesen
 A013   20 39 A0    JSR   INC$20
 A016   20 39 A0    JSR   INC$20
 A019   20 B5 80    JSR   SFACASC    ;FAC in Text umwandeln
 A01C   20 40 A0    JSR   OUT=       ;= ausgeben
 A01F   20 2B A0    JSR   OUTASC     ;Wert ausgeben
 A022   20 39 A0    JSR   INC$20
 A025   20 39 A0    JSR   INC$20
 A028   4C 47 9F    JMP   $9F47      ;zur naechsten Variablen

OUTASC:                              ;String ab $0100 ausgeben
 A02B   A2 00       LDX   #$00
 A02D   BD 00 01    LDA   $0100,X    ;Zeichen aus String
 A030   F0 06       BEQ   $A038      ;Endezeichen ?
 A032   20 D2 FF    JSR   BSOUT      ;Zeichen ausgeben
 A035   E8          INX              ;naechstes Zeichen
```

```
 A036   D0 F5       BNE   $A02D      ;Schleife
 A038   60          RTS

INC$20:                              ;$20,$21 um 1 erhoehen
 A039   E6 20       INC   $20
 A03B   D0 02       BNE   $A03F
 A03D   E6 21       INC   $21
 A03F   60          RTS

OUT=:
 A040   A9 3D       LDA   #$3D       ;"="
 A042   4C D2 FF    JMP   BSOUT      ;ausgeben

CHAREN1:
 A045   A5 01       LDA   $01        ;Prozessorport
 A047   09 04       ORA   #$04       ;Zeichengenerator un-
 A049   85 01       STA   $01        ;lesbar (Normalzustand)
 A04B   60          RTS

CHAREN0:
 A04C   A5 01       LDA   $01
 A04E   78          SEI              ;Interrupts verhindern
 A04F   29 FB       AND   #$FB       ;Zeichengen. lesbar
 A051   85 01       STA   $01
 A053   60          RTS

 A054   4C 2A 82    JMP   ENDSMB     ;Befehl beenden

BEFDRAW:
 A057   20 DB 83    JSR   INCBASBZ   ;Code ueberlesen
 A05A   20 B7 86    JSR   SGETSTR    ;String holen
 A05D   86 AC       STX   $AC        ;Stringadr. nach $AC,$AD
 A05F   85 AD       STA   $AD
 A061   A5 69       LDA   $69        ;Stringlaenge
 A063   48          PHA              ;retten
 A064   20 31 80    JSR   SCHKCOM    ;Komma ?

 A067   20 27 94    JSR   SGETADR    ;X-Wert holen
 A06A   8C 6D CB    STY   DRX        ;speichern
 A06D   8D 6E CB    STA   DRX+1
 A070   20 FC 81    JSR   SGETBYTC   ;Y-Wert
 A073   8E A3 C5    STX   DRY
 A076   20 7D 93    JSR   GETPKTF    ;Punktfarbe
 A079   68          PLA              ;Stringlaenge
 A07A   85 5F       STA   $5F        ;nach $5F
 A07C   A9 00       LDA   #$00
 A07E   8D 4F CB    STA   DRSTRZ     ;Zaehler im String auf 0
 A081   AD 02 C5    LDA   ROTGR      ;Groesse
 A084   8D 03 C5    STA   DRGRZ      ;Vergroesserungszaehler

 A087   8C 50 CB    STY   DRSTATUS
```

```
A08A  8E 51 CB   STX  DRYANF
A08D  AC 4F CB   LDY  DRSTRZ
A090  B1 AC      LDA  ($AC),Y    ;Zeichen aus String
A092  C4 5F      CPY  $5F        ;Stringlaenge erreicht ?
A094  F0 BE      BEQ  $A054      ;ja, dann Ende
A096  C8         INY
A097  18         CLC
A098  E9 2F      SBC  #$2F       ;$30 abziehen (Carry=0)
A09A  48         PHA             ;Wert von Ziffer retten
A09B  CE 03 C5   DEC  DRGRZ
A09E  AD 03 C5   LDA  DRGRZ
A0A1  D0 09      BNE  $A0AC      ;Vergr.-Z. nicht abgel. ?
A0A3  AD 02 C5   LDA  ROTGR
A0A6  8D 03 C5   STA  DRGRZ      ;Vergr.-Z. neu gesetzt
A0A9  8C 4F CB   STY  DRSTRZ
A0AC  AE 51 CB   LDX  DRYANF
A0AF  68         PLA             ;Ziffernwert
A0B0  AC 50 CB   LDY  DRSTATUS
A0B3  C9 09      CMP  #$09
A0B5  B0 9D      BCS  $A054      ;groesser als 8, dann Ende
A0B7  8D 61 CB   STA  DRRICHT    ;als Richtung merken
A0BA  C9 05      CMP  #$05
A0BC  90 03      BCC  $A0C1      ;kleiner als 5 ?
A0BE  18         CLC
A0BF  E9 04      SBC  #$04       ;5 abziehen
A0C1  AA         TAX             ;als Index
A0C2  BD 59 CB   LDA  DRAWTABX,X ;Increment aus Tabelle
A0C5  8D 65 CB   STA  DRINCX
A0C8  BD 5D CB   LDA  DRAWTABY,X
A0CB  8D 69 CB   STA  DRINCY
A0CE  AD 65 CB   LDA  DRINCX
A0D1  F0 1A      BEQ  $A0ED      ;Inc.X = 0 ?
A0D3  C9 FF      CMP  #$FF
A0D5  D0 0E      BNE  $A0E5      ;Inc.X nicht $FF ?

A0D7  AD 6D CB   LDA  DRX        ;DRX vermindern
A0DA  D0 03      BNE  $A0DF
A0DC  CE 6E CB   DEC  DRX+1
A0DF  CE 6D CB   DEC  DRX
A0E2  4C ED A0   JMP  $A0ED      ;weiter

A0E5  EE 6D CB   INC  DRX        ;DRX erhoehen
A0E8  D0 03      BNE  $A0ED
A0EA  EE 6E CB   INC  DRX+1

A0ED  AD 69 CB   LDA  DRINCY
A0F0  F0 0D      BEQ  $A0FF      ;Inc.Y = 0 ?
A0F2  C9 FF      CMP  #$FF
A0F4  D0 06      BNE  $A0FC      ;Inc.Y nicht $FF ?
```

```
A0F6   CE A3 C5   DEC  DRY        ;DRY vermindern
A0F9   4C FF A0   JMP  $A0FF      ;weiter

A0FC   EE A3 C5   INC  DRY        ;DRY erhoehen

A0FF   AD 6D CB   LDA  DRX        ;DRX und DRY als Punkt-
A102   85 09      STA  $09        ;Koord. $09,$0A und $A4
A104   AD 6E CB   LDA  DRX+1
A107   85 0A      STA  $0A
A109   AD A3 C5   LDA  DRY
A10C   85 A4      STA  $A4
A10E   AD 61 CB   LDA  DRRICHT    ;Wert von Ziffer
A111   C9 05      CMP  #$05
A113   90 06      BCC  $A11B      ;kleiner als 5 ?
A115   20 97 92   JSR  PUNKT      ;Punkt setzen
A118   20 0E 94   JSR  KERROMEIN   ;Kernal ein
A11B   4C 87 A0   JMP  $A087      ;zur Schleife

TABROT:
A11E   01 00 00 FF 00 FF 01 00    ;8 Byte pro Drehwert
A126   01 01 FF FF 01 FF 01 FF
A12E   00 01 FF 00 01 00 00 FF
A136   FF 01 FF 01 01 01 FF FF
A13E   FF 00 00 01 00 01 FF 00
A146   FF FF 01 01 FF 01 FF 01
A14E   00 FF 01 00 FF 00 00 01
A156   01 FF 01 FF FF FF 01 01

BEFROT:
A15E   20 02 82   JSR  SGETBYTN ;Drehwert holen
A161   E0 08      CPX  #$08
A163   90 05      BCC  $A16A      ;kleiner als 8 ?
A165   A9 00      LDA  #$00       ;Nr. f. 'bad mode'
A167   4C 8C 88   JMP  SERROUT    ;Fehler ausgeben

A16A   8A         TXA
A16B   0A         ASL  A
A16C   0A         ASL  A
A16D   0A         ASL  A
A16E   AA         TAX             ;X = Drehwert * 8
A16F   A0 00      LDY  #$00
A171   BD 1E A1   LDA  TABROT,X    ;8 Byte aus TABROT
A174   99 59 CB   STA  DRAWTABX,Y ;nach DRAWTAB uebertr.
A177   E8         INX
A178   C8         INY
A179   C0 08      CPY  #$08
A17B   30 F4      BMI  $A171       ;Schleife
A17D   20 FC 81   JSR  SGETBYTC    ;Groesse holen
A180   8E 02 C5   STX  ROTGR
A183   4C 2A 82   JMP  ENDSMB      ;Befehl fertig
```

```
BEFCHAR:
 A186   20 DB 83    JSR   INCBASBZ ;Code ueberlesen
 A189   20 27 94    JSR   SGETADR  ;X-K. holen
 A18C   84 09       STY   $09
 A18E   85 0A       STA   $0A
 A190   20 FC 81    JSR   SGETBYTC ;Y-K. holen
 A193   86 A4       STX   $A4
 A195   20 31 80    JSR   SCHKCOM  ;Komma ?
 A198   20 27 94    JSR   SGETADR  ;BS-Code holen
 A19B   84 AC       STY   $AC      ;Code nach $AC,$AD
 A19D   85 AD       STA   $AD
 A19F   20 7D 93    JSR   GETPKTF  ;Punktfarbe
 A1A2   20 FC 81    JSR   SGETBYTC ;Groesse
 A1A5   E0 00       CPX   #$00
 A1A7   D0 02       BNE   $A1AB    ;nicht null ?
 A1A9   A2 01       LDX   #$01
 A1AB   8E 6E CB    STX   CHGR
 A1AE   20 B4 A1    JSR   CHARZEI  ;Zeichen malen
 A1B1   4C 2A 82    JMP   ENDSMB   ;Befehl fertig

CHARZEI:                           ;1 Zeichen malen
 A1B4   A9 00       LDA   #$00
 A1B6   8D 6D CB    STA   DRX      ;Zaehler (bis 8) auf 0
 A1B9   20 4C A0    JSR   CHARENO  ;Zeich.gen. lesbar
 A1BC   06 AC       ASL   $AC
 A1BE   26 AD       ROL   $AD
 A1C0   06 AC       ASL   $AC
 A1C2   26 AD       ROL   $AD
 A1C4   06 AC       ASL   $AC
 A1C6   26 AD       ROL   $AD      ;Code * 8
 A1C8   18          CLC
 A1C9   A9 00       LDA   #$00
 A1CB   65 AC       ADC   $AC
 A1CD   8D A8 C5    STA   EY      ;EY,EY+1 = Code * 8 + $D000
 A1D0   A9 D0       LDA   #$D0
 A1D2   65 AD       ADC   $AD
 A1D4   8D A9 C5    STA   EY+1    ;Adr. im Zeichg. in EY,EY+1
 A1D7   A9 00       LDA   #$00
 A1D9   85 AC       STA   $AC     ;Bitzaehler = 0
 A1DB   85 AD       STA   $AD     ;Vergroesserungszaehler
 A1DD   E6 AD       INC   $AD     ;=1
 A1DF   A5 09       LDA   $09     ;Koord. in AX,AX+1 und
 A1E1   8D 97 C5    STA   AX      ;AY sichern
 A1E4   A5 0A       LDA   $0A
 A1E6   8D 98 C5    STA   AX+1
 A1E9   A5 A4       LDA   $A4
 A1EB   8D AC C5    STA   AY

 A1EE   AC 6D CB    LDY   DRX     ;Nr. der Bitzeile d. Zei.
 A1F1   A5 AC       LDA   $AC     ;Zaehler fuer einzelne Bits
 A1F3   29 07       AND   #$07    ;Bitnr. von 0-7
```

```
 A1F5   AA          TAX
 A1F6   AD  A8  C5  LDA   EY
 A1F9   85  FE      STA   $FE
 A1FB   AD  A9  C5  LDA   EY+1
 A1FE   85  FF      STA   $FF
 A200   B1  FE      LDA   ($FE),Y  ;Byte aus Zeichengen.
 A202   3D  30  93  AND   GBITTABS,X ;entspr. Bit maskieren
 A205   F0  06      BEQ   $A20D      ;Bit nicht gesetzt ?
 A207   20  97  92  JSR   PUNKT      ;Punkt setzen
 A20A   20  0E  94  JSR   KERROMEIN ;Kernal ein
 A20D   E6  AC      INC   $AC       ;naechstes Bit
 A20F   EE  97  C5  INC   AX        ;AX,AX+1 erhoehen
 A212   D0  03      BNE   $A217     ;(naechster Punkt in X)
 A214   EE  98  C5  INC   AX+1
 A217   A5  AC      LDA   $AC
 A219   C9  08      CMP   #$08
 A21B   D0  32      BNE   $A24F     ;Bitzaehler nicht abgel. ?
 A21D   A9  00      LDA   #$00
 A21F   85  AC      STA   $AC       ;Bitzaehler = 0
 A221   18          CLC
 A222   AD  97  C5  LDA   AX        ;AX,AX+1 um 8 vermindern,
 A225   E9  07      SBC   #$07      ;also auf urspr. Wert
 A227   8D  97  C5  STA   AX
 A22A   AD  98  C5  LDA   AX+1
 A22D   E9  00      SBC   #$00
 A22F   8D  98  C5  STA   AX+1
 A232   A5  AD      LDA   $AD       ;Vergr.Zae.
 A234   CD  6E  CB  CMP   CHGR
 A237   D0  07      BNE   $A240     ;nicht abgelaufen ?
 A239   EE  6D  CB  INC   DRX       ;Zaehler f. 8 Bit-Zeilen
 A23C   A9  00      LDA   #$00
 A23E   85  AD      STA   $AD
 A240   EE  AC  C5  INC   AY        ;naechste Zeile
 A243   E6  AD      INC   $AD
 A245   AD  6D  CB  LDA   DRX
 A248   C9  08      CMP   #$08      ;Zaehler bis 8 n. abgel. ?
 A24A   D0  03      BNE   $A24F
 A24C   4C  45  A0  JMP   CHAREN1 ;Zeichg. unlesb. und Ende

 A24F   AD  97  C5  LDA   AX        ;neue Koordinaten (AX,AY)
 A252   85  09      STA   $09       ;nach $09,$0A und $A4
 A254   AD  98  C5  LDA   AX+1
 A257   85  0A      STA   $0A
 A259   AD  AC  C5  LDA   AY
 A25C   85  A4      STA   $A4
 A25E   4C  EE  A1  JMP   $A1EE     ;Schleife

PTEST1:
 A261   84  90      STY   $90       ;ST = 0
 A263   B1  A8      LDA   ($A8),Y ;Byte aus Grafik-RAM
 A265   3D  30  93  AND   GBITTABS,X ;entspr. Bit
```

```
 A268  F0 03     BEQ  $A26D     ;Bit nicht gesetzt ?
 A26A  C8        INY
 A26B  84 90     STY  $90       ;sonst ST=1
 A26D  4C 0E 94  JMP  KERROMEIN ;Kernal ein und fertig

BEFHICOL:
 A270  A9 00     LDA  #$00
 A272  8D 1D CB  STA  LOWCOLFLAG ;Flag ruecksetzen
 A275  4C 74 91  JMP  BEFO      ;zum Leerbefehl

FILLBER:
 A278  A2 00     LDX  #$00      ;Zeilenzaehler
 A27A  A0 00     LDY  #$00      ;Spaltenzaehler
 A27C  CC 78 CB  CPY  SPALTENANZ
 A27F  F0 07     BEQ  $A288     ;Endwert erreicht ?
 A281  A5 66     LDA  $66       ;Fuellwert
 A283  91 20     STA  ($20),Y
 A285  C8        INY
 A286  D0 F4     BNE  $A27C     ;Schleife
 A288  E8        INX
 A289  EC 79 CB  CPX  ZEILENANZ
 A28C  F0 08     BEQ  $A296     ;Endwert erreicht ?
 A28E  A0 00     LDY  #$00
 A290  20 CE 94  JSR  20PL40    ;auf naechste Zeile
 A293  4C 7C A2  JMP  $A27C     ;Schleife

 A296  60        RTS            ;fertig

BEFFILL:
 A297  A9 00     LDA  #$00
 A299  85 A6     STA  $A6       ;Flag f. Farbe/Zeichen
 A29B  20 8D AC  JSR  GETRCWD   ;Grenzen holen
 A29E  A5 21     LDA  $21       ;Startadresse
 A2A0  48        PHA            ;sichern
 A2A1  A5 20     LDA  $20
 A2A3  48        PHA
 A2A4  20 FC 81  JSR  SGETBYTC  ;Zeichencode holen
 A2A7  86 66     STX  $66
 A2A9  20 78 A2  JSR  FILLBER   ;Video-RAM beschreiben
 A2AC  20 FC 81  JSR  SGETBYTC  ;Farbcode holen
 A2AF  86 66     STX  $66
 A2B1  38        SEC
 A2B2  68        PLA            ;alte Startadresse
 A2B3  E9 00     SBC  #$00      ;minus Video-RAM Startadr.
 A2B5  85 20     STA  $20
 A2B7  68        PLA
 A2B8  ED 88 02  SBC  VIDRAMHI
 A2BB  85 21     STA  $21
 A2BD  18        CLC
 A2BE  A5 20     LDA  $20
 A2C0  69 00     ADC  #$00      ;plus Farb-RAM-Startadr.
```

```
 A2C2   85 20      STA  $20
 A2C4   A5 21      LDA  $21
 A2C6   69 D8      ADC  #$D8
 A2C8   85 21      STA  $21        ;als neue Startadr.
 A2CA   20 78 A2   JSR  FILLBER    ;Farb-RAM beschreiben
 A2CD   4C 2A 82   JMP  ENDSMB     ;Befehl fertig

BEFFCHR:
 A2D0   A9 00      LDA  #$00       ;Flag f. Video-RAM
 A2D2   85 A6      STA  $A6
 A2D4   20 8D AC   JSR  GETRCWD    ;Grenzen holen
 A2D7   20 FC 81   JSR  SGETBYTC   ;Zeichencode holen
 A2DA   86 66      STX  $66
 A2DC   20 78 A2   JSR  FILLBER    ;Video-RAM beschreiben
 A2DF   4C 2A 82   JMP  ENDSMB     ;Befehl fertig

BEFFCOL:
 A2E2   A9 0A      LDA  #$0A       ;Flag f. Farb-RAM
 A2E4   85 A6      STA  $A6
 A2E6   20 8D AC   JSR  GETRCWD    ;Grenzen holen
 A2E9   20 FC 81   JSR  SGETBYTC   ;Farbcode holen
 A2EC   86 66      STX  $66
 A2EE   20 78 A2   JSR  FILLBER    ;Farb-RAM beschreiben
 A2F1   4C 2A 82   JMP  ENDSMB     ;Befehl fertig

BEFLOCAL:
 A2F4   A2 00      LDX  #$00       ;Zeiger in Local-Tab.
 A2F6   20 DB 83   JSR  INCBASBZ   ;Code ueberlesen
 A2F9   8A         TXA
 A2FA   48         PHA
 A2FB   20 BE 80   JSR  SGETVAR    ;Variable suchen
 A2FE   68         PLA
 A2FF   AA         TAX
 A300   A5 5F      LDA  $5F          ;$5F,$60 sind Var.adr. -2
 A302   9D 53 CA   STA  LOCALTAB,X ;in Tabelle legen
 A305   E8         INX
 A306   A5 60      LDA  $60
 A308   9D 53 CA   STA  LOCALTAB,X
 A30B   E8         INX
 A30C   A0 00      LDY  #$00
 A30E   B1 5F      LDA  ($5F),Y     ;Zeichen aus Var.name
 A310   9D 53 CA   STA  LOCALTAB,X ;in Tabelle legen
 A313   E8         INX
 A314   C8         INY
 A315   C0 02      CPY  #$02       ;2 Zeichen
 A317   D0 F5      BNE  $A30E      ;Schleife
 A319   A0 00      LDY  #$00
 A31B   A9 FF      LDA  #$FF       ;Merker f. unterdrueckte
 A31D   91 5F      STA  ($5F),Y    ;Variable setzen
 A31F   B1 7A      LDA  ($7A),Y    ;akt. Basic-Code
 A321   F0 0D      BEQ  $A330      ;Zeilenende ?
```

```
 A323   20 DB 83    JSR   INCBASBZ ;Trennzeichen ueberlesen
 A326   B1 7A       LDA   ($7A),Y
 A328   C9 2C       CMP   #$2C      ;folgt dann noch ein Komma
 A32A   D0 CD       BNE   $A2F9     ;nein, dann naechste Var.
 A32C   C9 20       CMP   #$20      ;Leerzeichen
 A32E   F0 F3       BEQ   $A323     ;ja, dann Schleife
 A330   A9 00       LDA   #$00      ;sonst Abschluss:
 A332   9D 53 CA    STA   LOCALTAB,X ;Localtabelle mit zwei
 A335   E8          INX
 A336   9D 53 CA    STA   LOCALTAB,X ;Nullen abschliessen
 A339   4C 2A 82    JMP   ENDSMB    ;Befehl fertig

BEFGLOBAL:
 A33C   A2 00       LDX   #$00       ;Zeiger in Local-Tabelle
 A33E   BD 53 CA    LDA   LOCALTAB,X ;Variablenadresse
 A341   85 5F       STA   $5F        ;nach $59,$60
 A343   E8          INX
 A344   BD 53 CA    LDA   LOCALTAB,X
 A347   85 60       STA   $60
 A349   D0 03       BNE   $A34E      ;nicht Ende der Tabelle ?
 A34B   4C 74 91    JMP   BEFO       ;zum Leerbefehl

 A34E   E8          INX
 A34F   A0 00       LDY   #$00
 A351   BD 53 CA    LDA   LOCALTAB,X ;Variablenname
 A354   91 5F       STA   ($5F),Y
 A356   E8          INX
 A357   C8          INY
 A358   C0 02       CPY   #$02      ;2 Zeichen
 A35A   D0 F5       BNE   $A351     ;Schleife
 A35C   F0 E0       BEQ   $A33E     ;fertig, dann naechst. V.

LINZEI:
 A35E   18          CLC
 A35F   AD 97 C5    LDA   AX
 A362   ED 2D C5    SBC   XMAXLOW
 A365   AD 98 C5    LDA   AX+1
 A368   ED 2E C5    SBC   XMAXHIGH ;(AX,AX+1) - XMAX - 1
 A36B   90 0C       BCC   $A379     ;kleiner 0 ?
 A36D   AD 2D C5    LDA   XMAXLOW   ;sonst AX=XMAX
 A370   8D 97 C5    STA   AX
 A373   AD 2E C5    LDA   XMAXHIGH
 A376   8D 98 C5    STA   AX+1
 A379   18          CLC
 A37A   AD AA C5    LDA   EX
 A37D   ED 2D C5    SBC   XMAXLOW
 A380   AD AB C5    LDA   EX+1
 A383   ED 2E C5    SBC   XMAXHIGH ;(EX,EX+1) - XMAX - 1
 A386   90 0C       BCC   $A394     ;kleiner 0 ?
 A388   AD 2D C5    LDA   XMAXLOW   ;sonst EX=XMAX
 A38B   8D AA C5    STA   EX
```

```
A38E  AD 2E C5  LDA  XMAXHIGH
A391  8D AB C5  STA  EX+1
A394  AD AC C5  LDA  AY
A397  C9 C7     CMP  #$C7
A399  90 05     BCC  $A3A0     ;AY kleiner 199 ?
A39B  A9 C7     LDA  #$C7      ;sonst AY=199
A39D  8D AC C5  STA  AY
A3A0  AD A8 C5  LDA  EY
A3A3  C9 C7     CMP  #$C7
A3A5  90 05     BCC  $A3AC     ;EY kleiner 199 ?
A3A7  A9 C7     LDA  #$C7      ;sonst EY=199
A3A9  8D A8 C5  STA  EY

MOBBEWEG: (allgemein: 2 Punkt mit Linie verbinden)
A3AC  A9 00     LDA  #$00
A3AE  8D 9F C5  STA  YDIFF     ;Hilszellen annullieren
A3B1  8D AD C5  STA  AY+1
A3B4  8D A9 C5  STA  EY+1
A3B7  8D A0 C5  STA  YDIFF+1
A3BA  8D A1 C5  STA  DIFFV
A3BD  8D A2 C5  STA  DIFFV+1

A3C0  AD 97 C5  LDA  AX        ;Punktkoordinaten setzen
A3C3  85 09     STA  $09
A3C5  AD 98 C5  LDA  AX+1
A3C8  85 0A     STA  $0A
A3CA  AD AC C5  LDA  AY
A3CD  85 A4     STA  $A4
A3CF  AD AD C5  LDA  AY+1
A3D2  85 A5     STA  $A5

A3D4  AD B4 C5  LDA  MOBBEW   ;Sprite bew. od. Lin. Zei.?
A3D7  C9 F0     CMP  #$F0
A3D9  D0 06     BNE  $A3E1    ;kein Sprite ?
A3DB  20 25 96  JSR  MOBPOSS  ;Sprite-Position setzen
A3DE  4C E4 A3  JMP  $A3E4    ;weiter
A3E1  20 97 92  JSR  PUNKT    ;Punkt setzen

A3E4  AD A1 C5  LDA  DIFFV
A3E7  0D A2 C5  ORA  DIFFV+1
A3EA  F0 06     BEQ  $A3F2    ;DIFFV = 0 ?
A3EC  20 0C A5  JSR  YNEXT    ;naechsten Punkt anpeilen
A3EF  4C C0 A3  JMP  $A3C0    ;zur Schleife

A3F2  AD 98 C5  LDA  AX+1     ;AX mit EX vergleichen
A3F5  8D 24 C5  STA  VGLA+1   ;Vergleichswert A
A3F8  AD 97 C5  LDA  AX
A3FB  8D 23 C5  STA  VGLA
A3FE  AD AB C5  LDA  EX+1
A401  8D 1B C5  STA  VGLE+1   ;Vergleichswert E
A404  AD AA C5  LDA  EX
```

```
A407  8D 1A C5   STA  VGLE
A40A  20 16 A6   JSR  VGLAE    ;Vergleichsroutine
A40D  D0 24      BNE  $A433    ;AX nicht gleich EX ?

A40F  AD AD C5   LDA  AY+1     ;AY mit EY vergleichen
A412  8D 24 C5   STA  VGLA+1
A415  AD AC C5   LDA  AY
A418  8D 23 C5   STA  VGLA
A41B  AD A9 C5   LDA  EY+1
A41E  8D 1B C5   STA  VGLE+1
A421  AD A8 C5   LDA  EY
A424  8D 1A C5   STA  VGLE
A427  20 16 A6   JSR  VGLAE    ;Vergleichsroutine
A42A  F0 06      BEQ  $A432    ;AY=EY, dann fertig
A42C  20 0C A5   JSR  YNEXT    ;naechsten Punkt anpeilen
A42F  4C C0 A3   JMP  $A3C0    ;zur Schleife

A432  60         RTS           ;fertig

A433  38         SEC
A434  AD A8 C5   LDA  EY       ;EY-AY bilden
A437  ED AC C5   SBC  AY
A43A  A8         TAY
A43B  AD A9 C5   LDA  EY+1
A43E  ED AD C5   SBC  AY+1
A441  48         PHA           ;EY-AY auf Stack
A442  98         TYA
A443  48         PHA
A444  B0 13      BCS  $A459    ;EY groesser gleich AY ?
A446  68         PLA
A447  68         PLA           ;Stack bereinigen
A448  38         SEC
A449  AD AC C5   LDA  AY       ;AY-EY bilden
A44C  ED A8 C5   SBC  EY
A44F  A8         TAY
A450  AD AD C5   LDA  AY+1
A453  ED A9 C5   SBC  EY+1
A456  48         PHA           ;AY-EY auf Stack
A457  98         TYA
A458  48         PHA

A459  18         CLC        ;hier ist abs(EY-AY) auf Stack
A45A  68         PLA
A45B  6D 9F C5   ADC  YDIFF
A45E  8D 9F C5   STA  YDIFF  ;YDIFF = YDIFF + abs(EY-AY)
A461  68         PLA
A462  6D A0 C5   ADC  YDIFF+1
A465  8D A0 C5   STA  YDIFF+1

A468  38         SEC
A469  AD AA C5   LDA  EX     ;EX-AX bilden
```

```
A46C  ED 97 C5   SBC  AX
A46F  A8         TAY
A470  AD AB C5   LDA  EX+1
A473  ED 98 C5   SBC  AX+1
A476  48         PHA            ;EX-AX auf Stack
A477  98         TYA
A478  48         PHA
A479  B0 13      BCS  $A48E     ;EX groeseer gleich AX ?
A47B  68         PLA
A47C  68         PLA            ;Stack bereinigen
A47D  38         SEC
A47E  AD 97 C5   LDA  AX        ;AX-EX bilden
A481  ED AA C5   SBC  EX
A484  A8         TAY
A485  AD 98 C5   LDA  AX+1
A488  ED AB C5   SBC  EX+1
A48B  48         PHA            ;AX-EX auf Stack
A48C  98         TYA
A48D  48         PHA

A48E  68         PLA       ;hier ist abs(EX-AX) auf Stack
A48F  8D A6 C5   STA  XDIFF     ;XDIFF = abs(EX-AX)
A492  68         PLA
A493  8D A7 C5   STA  XDIFF+1
A496  A9 00      LDA  #$00
A498  8D A2 C5   STA  DIFFV+1   ;DIFFV = 0
A49B  8D A1 C5   STA  DIFFV
A49E  38         SEC
A49F  AD 9F C5   LDA  YDIFF     ;YDIFF-XDIFF bilden
A4A2  ED A6 C5   SBC  XDIFF
A4A5  A8         TAY
A4A6  AD A0 C5   LDA  YDIFF+1
A4A9  ED A7 C5   SBC  XDIFF+1
A4AC  48         PHA            ;YDIFF-XDIFF auf Stack
A4AD  98         TYA
A4AE  48         PHA
A4AF  90 16      BCC  $A4C7     ;YDIFF kleiner XDIFF ?
A4B1  68         PLA
A4B2  8D 9F C5   STA  YDIFF     ;YDIFF = YDIFF - XDIFF
A4B5  68         PLA
A4B6  8D A0 C5   STA  YDIFF+1
A4B9  EE A1 C5   INC  DIFFV     ;DIFFV erhoehen (Zaehler)
A4BC  D0 03      BNE  $A4C1
A4BE  EE A2 C5   INC  DIFFV+1
A4C1  AD A1 C5   LDA  DIFFV
A4C4  4C 9B A4   JMP  $A49B     ;Schleife

A4C7  68         PLA
A4C8  68         PLA            ;Stack bereinigen
A4C9  AD 98 C5   LDA  AX+1      ;AX mit EX vergleichen
A4CC  8D 24 C5   STA  VGLA+1
```

```
 A4CF   AD 97 C5    LDA   AX
 A4D2   8D 23 C5    STA   VGLA
 A4D5   AD AB C5    LDA   EX+1
 A4D8   8D 1B C5    STA   VGLE+1
 A4DB   AD AA C5    LDA   EX
 A4DE   8D 1A C5    STA   VGLE
 A4E1   20 16 A6    JSR   VGLAE       ;Vergleichsroutine
 A4E4   F0 20       BEQ   $A506       ;gleich, dann zur Schleife
 A4E6   90 0E       BCC   $A4F6       ;AX kleiner EX ?

 A4E8   AD 97 C5    LDA   AX          ;AX vermindern
 A4EB   D0 03       BNE   $A4F0
 A4ED   CE 98 C5    DEC   AX+1
 A4F0   CE 97 C5    DEC   AX
 A4F3   4C FE A4    JMP   $A4FE

 A4F6   EE 97 C5    INC   AX          ;AX erhoehen
 A4F9   D0 03       BNE   $A4FE
 A4FB   EE 98 C5    INC   AX+1

 A4FE   AD A1 C5    LDA   DIFFV
 A501   0D A2 C5    ORA   DIFFV+1
 A504   D0 03       BNE   $A509       ;DIFFV nicht 0 ?
 A506   4C C0 A3    JMP   $A3C0       ;zur Schleife

 A509   4C EC A3    JMP   $A3EC       ;zur Schleife (ohne einen
                                      ;Punkt zu setzen)

YNEXT:
 A50C   AD A1 C5    LDA   DIFFV
 A50F   0D A2 C5    ORA   DIFFV+1
 A512   F0 0B       BEQ   $A51F       ;DIFFV = 0 ?
 A514   AD A1 C5    LDA   DIFFV       ;DIFFV vermindern
 A517   D0 03       BNE   $A51C
 A519   CE A2 C5    DEC   DIFFV+1
 A51C   CE A1 C5    DEC   DIFFV
 A51F   AD AD C5    LDA   AY+1        ;AY mit EY vergleichen
 A522   8D 24 C5    STA   VGLA+1
 A525   AD AC C5    LDA   AY
 A528   8D 23 C5    STA   VGLA
 A52B   AD A9 C5    LDA   EY+1
 A52E   8D 1B C5    STA   VGLE+1
 A531   AD A8 C5    LDA   EY
 A534   8D 1A C5    STA   VGLE
 A537   20 16 A6    JSR   VGLAE       ;Vergleichsroutine
 A53A   F0 16       BEQ   $A552       ;gleich, dann hier fertig
 A53C   B0 09       BCS   $A547       ;AY groesser als EY ?

 A53E   EE AC C5    INC   AY          ;AY erhoehen
 A541   D0 03       BNE   $A546
 A543   EE AD C5    INC   AY+1
```

```
A546   60         RTS

A547   AD AC C5   LDA   AY          ;AY vermindern
A54A   D0 03      BNE   $A54F
A54C   CE AD C5   DEC   AY+1
A54F   CE AC C5   DEC   AY
A552   60         RTS

A553   A5 A6      LDA   $A6         ;neue Zeilennr. (v. FIND)
A555   C5 A8      CMP   $A8         ;alte Zeilennr.
A557   D0 06      BNE   $A55F       ;ungleich ?
A559   A5 A7      LDA   $A7
A55B   C5 A9      CMP   $A9
A55D   F0 2A      BEQ   $A589       ;gleich, dann hier fertig
A55F   A5 A6      LDA   $A6         ;alte Zeil.nr. = neue Znr.
A561   85 A8      STA   $A8
A563   85 63      STA   $63
A565   A5 A7      LDA   $A7
A567   85 62      STA   $62
A569   85 A9      STA   $A9
A56B   A2 90      LDX   #$90
A56D   38         SEC
A56E   20 88 80   JSR   SXFLP        ;nach Fliessk. wandeln
A571   20 B5 80   JSR   SFACASC      ;in Text umwandeln
A574   20 2B A0   JSR   OUTASC       ;Text ausgeben
A577   38         SEC
A578   20 F0 FF   JSR   $FFF0        ;Cursor-Pos. holen
A57B   98         TYA                ;Cursorspalte
A57C   E9 0A      SBC   #$0A         ;minus 10
A57E   B0 FC      BCS   $A57C        ;noch groesser 0 ?
A580   49 FF      EOR   #$FF         ;invertieren
A582   69 01      ADC   #$01         ;+1 (also 1 - ...)
A584   AA         TAX                ;Zaehler f. Leerzeichen
A585   E8         INX
A586   CA         DEX                ;Zaehler vermindern
A587   D0 03      BNE   SPOUT        ;nicht 0, dann " " ausg.
A589   4C BB A5   JMP   $A5BB        ;weiter in Suchschleife

A58C   A9 20      LDA   #$20         ;Leerzeichen
A58E   20 D2 FF   JSR   BSOUT        ;ausgeben
A591   4C 86 A5   JMP   $A586        ;s.o.

BEFFIND:
A594   20 DB 83   JSR   INCBASBZ     ;Code ueberlesen
A597   A9 FF      LDA   #$FF
A599   85 A8      STA   $A8          ;Merker f. alte Zeilennr.
A59B   85 A9      STA   $A9
A59D   A0 00      LDY   #$00
A59F   A5 2B      LDA   $2B          ;$20,$21 mit Basic-Anfang
A5A1   85 20      STA   $20          ;($2B,$2C) laden
A5A3   A5 2C      LDA   $2C
```

```
A5A5   85 21      STA  $21
A5A7   4C DE A5   JMP  $A5DE      ;in die Suchschleife

A5AA   B1 7A      LDA  ($7A),Y    ;akt. Zeichen nach FIND
A5AC   F0 A5      BEQ  $A553      ;Zeilenende ?, dann gef.
A5AE   B1 20      LDA  ($20),Y    ;momentan gepr. Zeichen
A5B0   F0 09      BEQ  $A5BB      ;Zeilenende ?,dann n. gef.
A5B2   B1 7A      LDA  ($7A),Y
A5B4   D1 20      CMP  ($20),Y
A5B6   D0 03      BNE  $A5BB      ;Zeichen ungleich ?
A5B8   C8         INY             ;naechstes Zeichen
A5B9   D0 EF      BNE  $A5AA      ;Schleife
A5BB   B1 20      LDA  ($20),Y    ;mom. Zeichen
A5BD   C9 22      CMP  #$22       ;"
A5BF   D0 03      BNE  $A5C4      ;nein ?
A5C1   4C 03 A6   JMP  $A603      ;s.u.

A5C4   20 DC 9C   JSR  INC$20     ;naechstes Zeichen
A5C7   A0 00      LDY  #$00
A5C9   B1 20      LDA  ($20),Y
A5CB   D0 DD      BNE  $A5AA      ;kein Zeilenende ?
A5CD   20 DC 9C   JSR  INC$20     ;naechstes Zeichen
A5D0   A0 00      LDY  #$00
A5D2   8C DD C5   STY  PENDFLAG ;Prog.-Ende-Flag = 0
A5D5   B1 20      LDA  ($20),Y
A5D7   D0 05      BNE  $A5DE      ;nicht 0-Byte ?
A5D9   A9 01      LDA  #$01
A5DB   8D DD C5   STA  PENDFLAG ;Programm evtl. zu Ende
A5DE   20 DC 9C   JSR  INC$20
A5E1   B1 20      LDA  ($20),Y
A5E3   D0 0A      BNE  $A5EF      ;kein 0-Byte ?,dann weiter
A5E5   AD DD C5   LDA  PENDFLAG
A5E8   C9 01      CMP  #$01
A5EA   D0 03      BNE  $A5EF      ;kein Programmende ?
A5EC   4C 86 E3   JMP  WARM       ;Zum Warmstart; fertig

A5EF   20 DC 9C   JSR  INC$20     ;naechstes Zeichen
A5F2   B1 20      LDA  ($20),Y    ;Zeilennummer low
A5F4   85 A6      STA  $A6
A5F6   20 DC 9C   JSR  INC$20
A5F9   B1 20      LDA  ($20),Y    ;Zeilennummer high
A5FB   85 A7      STA  $A7
A5FD   20 DC 9C   JSR  INC$20
A600   4C AA A5   JMP  $A5AA      ;Zeilennummer ausgeben

A603   20 DC 9C   JSR  INC$20     ;naechstes Zeichen
A606   B1 20      LDA  ($20),Y
A608   C9 22      CMP  #$22       ;"
A60A   F0 07      BEQ  $A613      ;ja, dann weiter
A60C   C9 00      CMP  #$00
A60E   D0 F3      BNE  $A603    ;kein Zeil.ende ?, Schleife
```

```
 A610  4C C7 A5    JMP   $A5C7      ;zur naechsten Zeile
 A613  4C C4 A5    JMP   $A5C4      ;zur Suchschleife

VGLAE:       ;Vgl. VGLA mit VGLE; setzt Carry- und Zero-Flag
 A616  AD 23 C5    LDA   VGLA       ;1. Wert (low)
 A619  CD 1A C5    CMP   VGLE       ;2. Wert (low)
 A61C  F0 09       BEQ   $A627      ;gleich ?
 A61E  AD 24 C5    LDA   VGLA+1
 A621  ED 1B C5    SBC   VGLE+1     ;1.Wert - 2.Wert
 A624  09 01       ORA   #$01       ;Ungleichheit markieren
 A626  60          RTS

 A627  AD 24 C5    LDA   VGLA+1     ;1.Wert - 2.Wert
 A62A  ED 1B C5    SBC   VGLE+1
 A62D  60          RTS

BEFDESIGN:
 A62E  20 02 82    JSR   SGETBYTN ;Typ holen (0,1,2,3)
 A631  86 AA       STX   $AA
 A633  E0 04       CPX   #$04
 A635  90 05       BCC   $A63C      ;kleiner als 4 ?
 A637  A9 0C       LDA   #$0C       ;Nr. f. 'bad mode'
 A639  4C 8C 88    JMP   SERROUT    ;Fehler ausgeben

 A63C  A9 00       LDA   #$00
 A63E  8D 97 C5    STA   AX         ;Zeilenzaehler     = 0
 A641  8D AC C5    STA   AY         ;Zaehler in Zeile = 0
 A644  85 A6       STA   $A6        ;Byte-Zaehler      = 0
 A646  20 31 80    JSR   SCHKCOM    ;Komma ?
 A649  20 27 94    JSR   SGETADR    ;Adresse holen
 A64C  84 AE       STY   $AE        ;nach $AE,$AF
 A64E  85 AF       STA   $AF
 A650  A0 00       LDY   #$00
 A652  84 A8       STY   $A8        ;Sammelbyte
 A654  A2 00       LDX   #$00       ;Bitzaehler

 A656  B1 7A       LDA   ($7A),Y    ;akt. Code
 A658  F0 06       BEQ   $A660      ;Zeilenende ?
 A65A  20 DB 83    JSR   INCBASBZ ;vorruecken
 A65D  4C 56 A6    JMP   $A656      ;bis Zeilenende

 A660  20 DB 83    JSR   INCBASBZ ;Prg.Z. erhoehen
 A663  B1 7A       LDA   ($7A),Y    ;Vorw.z. low
 A665  8D 98 C5    STA   AX+1       ;speichern
 A668  20 DB 83    JSR   INCBASBZ ;naechstes Byte
 A66B  B1 7A       LDA   ($7A),Y    ;Vorw.Z. high
 A66D  0D 98 C5    ORA   AX+1
 A670  D0 05       BNE   $A677      ;Vorw.Z. nicht 0 ?

 A672  A9 0A       LDA   #$0A       ;Nr. F. 'too few lines'
 A674  4C 8C 88    JMP   SERROUT    ;Fehler ausgeben
```

```
 A677   20 DB 83    JSR   INCBASBZ ;Zeilennr. ueberlesen
 A67A   20 DB 83    JSR   INCBASBZ
 A67D   20 DB 83    JSR   INCBASBZ
 A680   A0 00       LDY   #$00
 A682   B1 7A       LDA   ($7A),Y  ;akt. Code
 A684   C9 40       CMP   #$40     ;Klammeraffe
 A686   F0 F5       BEQ   $A67D    ;ja ?,dann ueberlesen
 A688   A5 AA       LDA   $AA      ;Design-Typ
 A68A   F0 07       BEQ   $A693    ;0 ?
 A68C   C9 02       CMP   #$02
 A68E   F0 03       BEQ   $A693    ;2 ?
 A690   4C 22 A7    JMP   $A722    ;sonst zu $A722

;Design-Typ 0 und 2 (normale Grafik)
 A693   B1 7A       LDA   ($7A),Y  ;akt. Zeichen
 A695   C9 41       CMP   #$41     ;"A" ?
 A697   F0 18       BEQ   $A6B1
 A699   C9 2E       CMP   #$2E     ;"." ?
 A69B   F0 14       BEQ   $A6B1
 A69D   C9 20       CMP   #$20     ;" " ?
 A69F   F0 10       BEQ   $A6B1
 A6A1   C9 42       CMP   #$42     ;"B" ?
 A6A3   F0 05       BEQ   $A6AA
 A6A5   A9 0B       LDA   #$0B     ;Nr. f. 'bad char f a m'
 A6A7   4C 8C 88    JMP   SERROUT  ;Fehler ausgeben

 A6AA   A5 A8       LDA   $A8
 A6AC   1D 30 93    ORA   GBITTABS,X ;entspr. Bit setzen
 A6AF   85 A8       STA   $A8
 A6B1   E8          INX            ;Bitzaehler
 A6B2   E0 08       CPX   #$08
 A6B4   D0 C7       BNE   $A67D    ;nicht abgelaufen ?
 A6B6   A4 A6       LDY   $A6      ;Bytezaehler
 A6B8   A5 A8       LDA   $A8      ;Sammel-Byte
 A6BA   91 AE       STA   ($AE),Y  ;Ergebnis speichern
 A6BC   E6 A6       INC   $A6      ;Bytezaehler erh.
 A6BE   EE AC C5    INC   AY       ;Bytezaehler in Zeile erh.
 A6C1   A2 00       LDX   #$00     ;Bitzaehler = 0
 A6C3   86 A8       STX   $A8      ;Sammelbyte = 0
 A6C5   A5 AA       LDA   $AA      ;Design-Typ
 A6C7   C9 02       CMP   #$02
 A6C9   B0 07       BCS   $A6D2    ;groesser gleich 2 ?
 A6CB   AD AC C5    LDA   AY       ;Zaehler in Zeile
 A6CE   C9 03       CMP   #$03
 A6D0   D0 AB       BNE   $A67D    ;nicht 3, dann Schleife
 A6D2   A0 00       LDY   #$00
 A6D4   8C AC C5    STY   AY       ;Bytezaehler in Zeile = 0
 A6D7   EE 97 C5    INC   AX       ;Zeilenzaehler erhoehen
 A6DA   A5 AA       LDA   $AA      ;Design-Typ
 A6DC   C9 02       CMP   #$02
 A6DE   90 07       BCC   $A6E7    ;kleiner als 2 ?
```

```
 A6E0   AD 97 C5    LDA   AX         ;Zeilenzaehler
 A6E3   C9 08       CMP   #$08
 A6E5   F0 0A       BEQ   $A6F1      ;= 8 ?
 A6E7   AD 97 C5    LDA   AX
 A6EA   C9 15       CMP   #$15
 A6EC   F0 03       BEQ   $A6F1      ;= 21 ?
 A6EE   4C 50 A6    JMP   $A650      ;Schleife

 A6F1   B1 7A       LDA   ($7A),Y
 A6F3   F0 06       BEQ   $A6FB      ;Zeilenende, dann fertig
 A6F5   20 DB 83    JSR   INCBASBZ   ;naechstes Zeichen
 A6F8   4C F1 A6    JMP   $A6F1      ;Schleife

 A6FB   4C 2A 82    JMP   ENDSMB     ;Befehl fertig

UEBER4:                          ;4 Byte ueberlesen und fertig
 A6FE   20 DB 83    JSR   INCBASBZ
 A701   20 DB 83    JSR   INCBASBZ
 A704   20 DB 83    JSR   INCBASBZ
 A707   4C 74 91    JMP   BEFO

DESTAB3:
 A70A   C0 00 30 00 0C 00 03 00  ;beide Bits gesetzt

DESTAB2:
 A712   80 00 20 00 08 00 02 00  ;jeweils 1. Bit gesetzt

DESTAB1:
 A71A   40 00 10 00 04 00 01 00  ;jeweils 2. Bit gesetzt

;Design-Typ 1 und 3 (Multi-Colour)
 A722   A0 00       LDY   #$00
 A724   B1 7A       LDA   ($7A),Y    ;akt. Zeichen
 A726   C9 44       CMP   #$44       ;"D"
 A728   D0 0B       BNE   $A735      ;nein ?
 A72A   A5 A8       LDA   $A8        ;Sammelbyte
 A72C   1D 0A A7    ORA   DESTAB3,X  ;2 Bit setzen
 A72F   85 A8       STA   $A8
 A731   E8          INX              ;naechstes Bit(paar)
 A732   4C B1 A6    JMP   $A6B1      ;zur Schleife

 A735   C9 43       CMP   #$43       ;"C"
 A737   D0 0B       BNE   $A744      ;nein ?
 A739   A5 A8       LDA   $A8
 A73B   1D 12 A7    ORA   DESTAB2,X  ;vorderes Bit setzen
 A73E   85 A8       STA   $A8
 A740   E8          INX              ;naechstes Bit(paar)
 A741   4C B1 A6    JMP   $A6B1      ;zur Schleife

 A744   C9 42       CMP   #$42       ;"B"
 A746   D0 0B       BNE   $A753      ;nein ?
```

```
 A748   A5 A8       LDA  $A8
 A74A   1D 1A A7    ORA  DESTAB1,X ;hinteres Bit setzen
 A74D   85 A8       STA  $A8
 A74F   E8          INX            ;naechstes Bit(paar)
 A750   4C B1 A6    JMP  $A6B1     ;zur Schleife

 A753   C9 2E       CMP  #$2E      ;"."
 A755   F0 0D       BEQ  $A764     ;ja, dann weiter
 A757   C9 41       CMP  #$41      ;"A"
 A759   F0 09       BEQ  $A764     ;ja, dann weiter
 A75B   C9 20       CMP  #$20      ;" "
 A75D   F0 05       BEQ  $A764     ;ja, dann weiter
 A75F   A9 0B       LDA  #$0B      ;Nr. f. "bad char f a mob'
 A761   4C 8C 88    JMP  SERROUT   ;Fehler ausgeben

 A764   E8          INX            ;naechstes Bit(paar)
 A765   4C B1 A6    JMP  $A6B1     ;zur Schleife

BEFRLOCM:
 A768   20 02 82    JSR  SGETBYTN ;Sprite-Nr. holen
 A76B   8E BB C5    STX  MOBNR2
 A76E   8E BC C5    STX  MOBNR
 A771   0E BB C5    ASL  MOBNR2    ;doppelte Spsrite-Nr.
 A774   AD 10 D0    LDA  VICMX8    ;X-Koord.-Uebertraege
 A777   1D 78 97    ORA  MBITTABS,X
 A77A   CD 10 D0    CMP  VICMX8
 A77D   D0 08       BNE  $A787     ;war Bit nicht gesetzt ?
 A77F   A9 01       LDA  #$01
 A781   8D 98 C5    STA  AX+1      ;X-Koord. High = 1
 A784   4C 8C A7    JMP  $A78C
 A787   A9 00       LDA  #$00
 A789   8D 98 C5    STA  AX+1      ;X-Koord. High = 0
 A78C   AC BB C5    LDY  MOBNR2
 A78F   B9 00 D0    LDA  VICMX,Y   ;X-Koordinate (low)
 A792   8D 97 C5    STA  AX        ;als Anfangswert
 A795   B9 01 D0    LDA  VICMY,Y   ;Y-Koordinate
 A798   8D AC C5    STA  AY
 A79B   A9 00       LDA  #$00
 A79D   8D AD C5    STA  AY+1      ;Y-Koordinate (High)
 A7A0   20 6B 93    JSR  $936B     ;EX,EY und $F7 holen
 A7A3   4C E2 95    JMP  $95E2     ;in den MMOB-Befehl

BEFCMOB:
 A7A6   20 02 82    JSR  SGETBYTN ;1. Farbe holen
 A7A9   8E 25 D0    STX  VICMMC1   ;in VIC eintragen
 A7AC   20 FC 81    JSR  SGETBYTC ;2. Farbe holen
 A7AF   8E 26 D0    STX  VICMMC2
 A7B2   4C 2A 82    JMP  ENDSMB    ;Befehl fertig
```

```
BEFBCKGNDS:
 A7B5   AD 11 D0   LDA  VICST1    ;VIC Steuerregister 1
 A7B8   29 DF      AND  #$DF      ;Punktgrafik ausschalten
 A7BA   09 40      ORA  #$40      ;Mehrfarbigen Hi.gr. ein
 A7BC   8D 11 D0   STA  VICST1
 A7BF   AD 16 D0   LDA  VICST2    ;VIC Steuerregister 2
 A7C2   29 EF      AND  #$EF      ;Mehrfarbenmodus aus
 A7C4   8D 16 D0   STA  VICST2
 A7C7   20 02 82   JSR  SGETBYTN  ;Hintergrundfarbe 0 holen
 A7CA   8E 21 D0   STX  VICHIFAR
 A7CD   20 FC 81   JSR  SGETBYTC  ;Hintergrundfarbe 1 holen
 A7D0   8E 22 D0   STX  VICHIFAR+1
 A7D3   20 FC 81   JSR  SGETBYTC  ;Hintergrundfarbe 2 holen
 A7D6   8E 23 D0   STX  VICHIFAR+2
 A7D9   20 FC 81   JSR  SGETBYTC  ;Hintergrundfarbe 3 holen
 A7DC   8E 24 D0   STX  VICHIFAR+3
 A7DF   4C 2A 82   JMP  ENDSMB    ;Befehl fertig

BEFPAUSE:
 A7E2   20 73 00   JSR  CHRGET    ;Code ueberlesen
 A7E5   A0 00      LDY  #$00
 A7E7   B1 7A      LDA  ($7A),Y   ;akt. Basic-Code
 A7E9   C9 22      CMP  #$22      ;"
 A7EB   D0 23      BNE  $A810     ;kein Anf.zeichen ?
 A7ED   A5 7A      LDA  $7A       ;Prg.Z. vermindern
 A7EF   D0 02      BNE  $A7F3
 A7F1   C6 7B      DEC  $7B
 A7F3   C6 7A      DEC  $7A
 A7F5   20 B7 86   JSR  SGETSTR  ;String holen
 A7F8   85 A9      STA  $A9      ;Stringadresse nach $A8,$A9
 A7FA   86 A8      STX  $A8
 A7FC   A0 00      LDY  #$00
 A7FE   B1 A8      LDA  ($A8),Y  ;Zeichen aus String
 A800   20 D2 FF   JSR  BSOUT    ;ausgeben
 A803   C8         INY           ;naechstes Zeichen
 A804   C4 69      CPY  $69      ;Stringlaenge
 A806   D0 F6      BNE  $A7FE    ;nicht erreicht ?
 A808   A9 0D      LDA  #$0D     ;(CR)
 A80A   20 D2 FF   JSR  BSOUT    ;ausgeben
 A80D   20 31 80   JSR  SCHKCOM  ;Komma ?
 A810   20 27 94   JSR  SGETADR  ;Wert f. Pause holen
 A813   A9 00      LDA  #$00
 A815   8D 17 C5   STA  ZAESEC   ;Sekundenzaehler = 0
 A818   8D 16 C5   STA  ZAEIRQ   ;60stel-Sek.-Zae. = 0
 A81B   20 E4 FF   JSR  GETIN    ;Zeichen von Tastatur
 A81E   C9 0D      CMP  #$0D     ;RETURN-Taste
 A820   F0 0C      BEQ  $A82E    ;ja, dann Ende
 A822   AD 17 C5   LDA  ZAESEC   ;Sekundenzaehler
 A825   C5 65      CMP  $65      ;Sollwert
 A827   B0 05      BCS  $A82E    ;gleich, dann Ende
 A829   F0 03      BEQ  $A82E    ;(unsinning)
```

```
 A82B  4C 1B A8   JMP  $A81B     ;zur Warte-Schleife

 A82E  4C 2A 82   JMP  ENDSMB    ;Befehl fertig

BEFNRM:
 A831  20 37 A8   JSR  NRM       ;Normalmodus einschalten
 A834  4C 74 91   JMP  BEFO      ;zum Leerbefehl

NRM:
 A837  A9 1B      LDA  #$1B
 A839  8D 11 D0   STA  VICST1    ;VIC-Steuerreg.1 auf $1B
 A83C  A9 15      LDA  #$15
 A83E  8D 18 D0   STA  VICADS    ;VIC-Adressreg.  auf $15
 A841  A9 C8      LDA  #$C8
 A843  8D 16 D0   STA  VICST2    ;VIC-Steuerreg.2 auf $C8
 A846  A9 C7      LDA  #$C7
 A848  8D 00 DD   STA  CIA2PRA ;CIA2-Port A     auf $C7
 A84B  A9 63      LDA  #$63
 A84D  8D B3 C5   STA  GMEMFLAG ;Grafik-RAM-Flag
 A850  A9 04      LDA  #$04
 A852  8D 88 02   STA  VIDRAMHI ;Video-RAM nach $0400
 A855  60         RTS

BEFMOBOFF:
 A856  20 02 82   JSR  SGETBYTN ;Sprite-Nr. holen
 A859  AD 15 D0   LDA  VICMEA   ;Sprite-Ein/Aus-Register
 A85C  3D 80 97   AND  MBITTABL,X ;entspr. Bit loeschen
 A85F  8D 15 D0   STA  VICMEA
 A862  4C 2A 82   JMP  ENDSMB    ;Befehl fertig

BEFOFF:
 A865  A2 00      LDX  #$00
 A867  8E C6 C5   STX  FLASHFL   ;Flash-Flag ruecks.
 A86A  8A         TXA
 A86B  9D 04 C5   STA  FLASHFLS,X ;Flag pro Farbe loeschen
 A86E  E8         INX            ;naechste Farbe
 A86F  E0 10      CPX  #$10      ;16 Farben
 A871  90 F8      BCC  $A86B     ;Farbzaehler n. abgel. ?
 A873  4C 74 91   JMP  BEFO      ;zum Leerbefehl

BEFARC:
 A876  20 8D A8   JSR  ARC1
 A879  AD 18 C5   LDA  STRZ1     ;eff. Startwinkel
 A87C  85 A8      STA  $A8
 A87E  AD 19 C5   LDA  STRZ2
 A881  85 A9      STA  $A9
 A883  A9 0A      LDA  #$0A
 A885  85 21      STA  $21       ;Flag f. mehrere Punkte
 A887  20 4F A9   JSR  KREIS1    ;Kreis-Routine
 A88A  4C 2A 82   JMP  ENDSMB    ;Befehl fertig
```

```
ARC1:
 A88D   20 DB 83   JSR  INCBASBZ ;Code ueberl.
 A890   A9 00      LDA  #$00
 A892   8D 7C C5   STA  DREHSINN ;Flags ruecks.
 A895   8D 2F C5   STA  UEBERDREH
 A898   20 27 94   JSR  SGETADR  ;Mittelp. X-Koord. holen
 A89B   8C 4B CB   STY  KMX
 A89E   8D 4C CB   STA  KMX+1
 A8A1   20 31 80   JSR  SCHKCOM  ;Komma ?
 A8A4   20 27 94   JSR  SGETADR  ;Mittelp. Y-Koord. holen
 A8A7   8C 49 CB   STY  KMY
 A8AA   20 31 80   JSR  SCHKCOM  ;Komma ?
 A8AD   20 27 94   JSR  SGETADR  ;Startwinkel holen
 A8B0   84 A8      STY  $A8
 A8B2   85 A9      STA  $A9
 A8B4   20 31 80   JSR  SCHKCOM  ;Komma ?
 A8B7   20 27 94   JSR  SGETADR  ;Endwinkel holen
 A8BA   84 AC      STY  $AC
 A8BC   85 AD      STA  $AD
 A8BE   20 31 80   JSR  SCHKCOM  ;Komma ?
 A8C1   20 27 94   JSR  SGETADR  ;Winkel-Schrittw. holen
 A8C4   8D 4E CB   STA  WINKSW+1
 A8C7   8C 4D CB   STY  WINKSW
 A8CA   A5 A9      LDA  $A9      ;Anfangswinkel mit Endw.
 A8CC   8D 24 C5   STA  VGLA+1   ;vergleichen
 A8CF   A5 A8      LDA  $A8
 A8D1   8D 23 C5   STA  VGLA
 A8D4   A5 AD      LDA  $AD
 A8D6   8D 1B C5   STA  VGLE+1
 A8D9   A5 AC      LDA  $AC
 A8DB   8D 1A C5   STA  VGLE
 A8DE   20 16 A6   JSR  VGLAE    ;Vergleichsroutine
 A8E1   90 18      BCC  $A8FB    ;Anf.Wi. kleiner Endwi. ?
 A8E3   AD 7C C5   LDA  DREHSINN
 A8E6   49 0A      EOR  #$0A     ;Flag umdrehen
 A8E8   8D 7C C5   STA  DREHSINN
 A8EB   A6 AC      LDX  $AC      ;$A8,$A9 mit $AC,$AD
 A8ED   A4 AD      LDY  $AD      ;vertauschen
 A8EF   A5 A8      LDA  $A8
 A8F1   85 AC      STA  $AC
 A8F3   A5 A9      LDA  $A9
 A8F5   85 AD      STA  $AD
 A8F7   86 A8      STX  $A8
 A8F9   84 A9      STY  $A9
 A8FB   A5 AC      LDA  $AC      ;eff. Endwinkel merken
 A8FD   8D 18 C5   STA  STRZ1
 A900   A5 AD      LDA  $AD
 A902   8D 19 C5   STA  STRZ2
 A905   20 FC 81   JSR  SGETBYTC ;Radius X-Ri. holen
 A908   8E 48 CB   STX  KRX
 A90B   20 FC 81   JSR  SGETBYTC ;Radius Y-Ri. holen
```

```
 A90E  8E C5 CB    STX  KRY
 A911  20 7D 93    JSR  GETPKTF   ;Punktfarbe holen

KREIS:
 A914  A9 14       LDA  #$14      ;Flag f. mehrere Punkte
 A916  85 21       STA  $21
 A918  A5 A8       LDA  $A8       ;Startwinkel sw merken
 A91A  48          PHA
 A91B  A5 A9       LDA  $A9
 A91D  48          PHA
 A91E  20 D7 AA    JSR  A8A9DIV90 ;Startwinkel / 90
 A921  20 01 AA    JSR  SINMALKRX ;sin(sw) * KRX
 A924  A5 65       LDA  $65       ;Ergebnis nach $AA,$AB
 A926  85 AA       STA  $AA
 A928  A5 64       LDA  $64
 A92A  85 AB       STA  $AB
 A92C  20 1F AA    JSR  COSMALKRY ;cos(sw) * KRY
 A92F  A5 65       LDA  $65       ;Ergebnis nach $AE,$AF
 A931  85 AE       STA  $AE
 A933  A5 64       LDA  $64
 A935  85 AF       STA  $AF
 A937  20 71 AA    JSR  PKBER     ;Punktkooord. berechnen
 A93A  A5 AE       LDA  $AE       ;Ergebnis nach EX,EY
 A93C  8D A8 C5    STA  EY
 A93F  A5 AA       LDA  $AA
 A941  8D AA C5    STA  EX
 A944  A5 AB       LDA  $AB
 A946  8D AB C5    STA  EX+1
 A949  68          PLA
 A94A  85 A9       STA  $A9
 A94C  68          PLA
 A94D  85 A8       STA  $A8
KREIS1:
 A94F  A5 A8       LDA  $A8       ;Winkel auf Stapel legen
 A951  48          PHA
 A952  A5 A9       LDA  $A9
 A954  48          PHA
 A955  20 D7 AA    JSR  A8A9DIV90 ;Winkel / 90
 A958  20 01 AA    JSR  SINMALKRX ;sin(wi) * KRX
 A95B  84 AA       STY  $AA       ;Ergebnis nach $AA,$AB
 A95D  85 AB       STA  $AB
 A95F  20 1F AA    JSR  COSMALKRY ;cos(wi) * KRY
 A962  84 AE       STY  $AE       ;Ergebnis nach $AE,$AF
 A964  85 AF       STA  $AF
 A966  20 71 AA    JSR  PKBER     ;Punktkoord. berechnen
 A969  A5 AE       LDA  $AE       ;Ergebnis nach AX,AY
 A96B  8D AC C5    STA  AY
 A96E  A5 AA       LDA  $AA
 A970  8D 97 C5    STA  AX
 A973  A5 AB       LDA  $AB
 A975  8D 98 C5    STA  AX+1
```

```
A978  AD 97 C5  LDA  AX        ;AX,AX+1 und AY sichern
A97B  48        PHA
A97C  AD AC C5  LDA  AY
A97F  48        PHA
A980  AD 98 C5  LDA  AX+1
A983  48        PHA
A984  20 5E A3  JSR  LINZEI    ;Linie zeichnen
A987  20 0E 94  JSR  KERROMEIN ;Kernal ein
A98A  68        PLA         ;alte Werte v. AX,AY n. EX,EY
A98B  8D AB C5  STA  EX+1
A98E  68        PLA
A98F  8D A8 C5  STA  EY
A992  68        PLA
A993  8D AA C5  STA  EX
A996  68        PLA          ;alten Winkelwert n. $A8,$A9
A997  85 A9     STA  $A9
A999  68        PLA
A99A  85 A8     STA  $A8
A99C  A5 21     LDA  $21     ;Mehrfachflag
A99E  C9 14     CMP  #$14
A9A0  F0 01     BEQ  $A9A3   ;gesetzt ?
A9A2  60        RTS          ;sonst fertig

A9A3  AD 7C C5  LDA  DREHSINN ;Flag
A9A6  C9 0A     CMP  #$0A
A9A8  D0 21     BNE  $A9CB     ;nicht gesetzt ?
A9AA  38        SEC
A9AB  A5 A8     LDA  $A8    ;$A8,$A9 = $A8,$A9 - Wink.sw.
A9AD  ED 4D CB  SBC  WINKSW
A9B0  85 A8     STA  $A8
A9B2  A5 A9     LDA  $A9
A9B4  ED 4E CB  SBC  WINKSW+1
A9B7  85 A9     STA  $A9
A9B9  B0 1F     BCS  $A9DA
A9BB  A9 0A     LDA  #$0A
A9BD  8D 2F C5  STA  UEBERDREH ;Ueberlauf-Flag setzen
A9C0  A9 68     LDA  #$68      ;360 nach $A8,$A9
A9C2  85 A8     STA  $A8
A9C4  A9 01     LDA  #$01
A9C6  85 A9     STA  $A9
A9C8  4C DA A9  JMP  $A9DA     ;weiter s.u.

A9CB  18        CLC
A9CC  A5 A8     LDA  $A8    ;$A8,$A9 = $A8,$A9 + Wink.sw.
A9CE  6D 4D CB  ADC  WINKSW
A9D1  85 A8     STA  $A8
A9D3  A5 A9     LDA  $A9
A9D5  6D 4E CB  ADC  WINKSW+1
A9D8  85 A9     STA  $A9
A9DA  8D 24 C5  STA  VGLA+1
A9DD  A5 A8     LDA  $A8
```

```
 A9DF   8D 23 C5    STA   VGLA      ;$A8,$A9 und $AC,$AD als
 A9E2   A5 AD       LDA   $AD       ;als Vergleichsw. speich.
 A9E4   8D 1B C5    STA   VGLE+1
 A9E7   A5 AC       LDA   $AC
 A9E9   8D 1A C5    STA   VGLE
 A9EC   AD 7C C5    LDA   DREHSINN  ;Flag
 A9EF   C9 0A       CMP   #$0A
 A9F1   D0 03       BNE   $A9F6     ;nicht gesetzt ?
 A9F3   4C 57 AB    JMP   $AB57

 A9F6   20 16 A6    JSR   VGLAE     ;Vgl.-Routine
 A9F9   F0 03       BEQ   $A9FE     ;akt.Wi. = Endwi. ?
 A9FB   90 01       BCC   $A9FE     ;akt.Wi. kleiner Endwi. ?
 A9FD   60          RTS             ;sonst fertig

 A9FE   4C 4F A9    JMP   KREIS1    ;Schleife

SINMALKRX:
 AA01   20 3D AA    JSR   A8A9BOGM  ;Winkel n. Bogenmass
 AA04   20 1F 80    JSR   SSIN      ;Sinus ber.
 AA07   20 91 80    JSR   SFACARG   ;Sinus nach ARG
 AA0A   AD 48 CB    LDA   KRX       ;Radius X
 AA0D   85 63       STA   $63
 AA0F   A5 A9       LDA   $A9
 AA11   85 62       STA   $62
 AA13   A2 90       LDX   #$90
 AA15   38          SEC
 AA16   20 88 80    JSR   SXFLP     ;n. Fliessk.
 AA19   20 9A 80    JSR   SFMULT    ;mal Sinus
 AA1C   4C 16 80    JMP   SFACADR   ;Ergebnis n. Adressformat

COSMALKRY:
 AA1F   20 3D AA    JSR   A8A9BOGM  ;Winkel n. Bogenmass
 AA22   20 28 80    JSR   SCOS      ;Cosinus berechnen
 AA25   20 91 80    JSR   SFACARG   ;n. Argument uebertr.
 AA28   AD C5 CB    LDA   KRY       ;Radius Y
 AA2B   85 63       STA   $63
 AA2D   A5 A9       LDA   $A9
 AA2F   85 62       STA   $62
 AA31   A2 90       LDX   #$90
 AA33   38          SEC
 AA34   20 88 80    JSR   SXFLP     ;n. Fliessk. wandeln
 AA37   20 9A 80    JSR   SFMULT    ;mal ARG (Cosinus)
 AA3A   4C 16 80    JMP   SFACADR   ;Ergebnis n. Adr.form.

A8A9BOGM:
 AA3D   A5 A8       LDA   $A8       ;$A8,$A9
 AA3F   85 63       STA   $63
 AA41   A5 A9       LDA   $A9
 AA43   85 62       STA   $62
 AA45   A2 90       LDX   #$90
```

```
 AA47   38          SEC
 AA48   20 88 80    JSR   SXFLP     ;n. Fliessk.
 AA4B   20 91 80    JSR   SFACARG   ;n. ARG
 AA4E   A9 A8       LDA   #$A8      ;Adr. von Konstante PI
 AA50   A0 AE       LDY   #$AE
 AA52   20 A3 80    JSR   SFLDFACK  ;FAC mit PI laden
 AA55   20 9A 80    JSR   SFMULT    ;mal $A8,$A9
 AA58   20 91 80    JSR   SFACARG   ;Erg. n. ARG
 AA5B   A9 B4       LDA   #$B4      ;Konst. 180
 AA5D   85 63       STA   $63
 AA5F   A9 00       LDA   #$00
 AA61   85 62       STA   $62
 AA63   A2 90       LDX   #$90
 AA65   38          SEC
 AA66   20 88 80    JSR   SXFLP     ;n. Fliessk.
 AA69   A5 61       LDA   $61
 AA6B   A4 70       LDY   $70
 AA6D   20 AC 80    JSR   SFDIV     ;FAC/180
 AA70   60          RTS             ;FAC = $A8/A9 * PI / 180

PKBER:                              ;absolute Punktkoord. ber.
 AA71   A5 20       LDA   $20       ;DIV(Winkel,90)
 AA73   C9 00       CMP   #$00
 AA75   D0 06       BNE   $AA7D     ;nicht 0 ?
 AA77   20 C8 AA    JSR   KMYMINUS
 AA7A   4C 97 AA    JMP   KMXPLUS

 AA7D   C9 01       CMP   #$01
 AA7F   D0 06       BNE   $AA87     ;nicht 1 ?
 AA81   20 A7 AA    JSR   KMYPLUS
 AA84   4C 97 AA    JMP   KMXPLUS

 AA87   C9 02       CMP   #$02
 AA89   D0 06       BNE   $AA91     ;nicht 2 ?
 AA8B   20 B0 AA    JSR   KMXMINUS
 AA8E   4C A7 AA    JMP   KMYPLUS

 AA91   20 B0 AA    JSR   KMXMINUS
 AA94   4C C8 AA    JMP   KMYMINUS

KMXPLUS:
 AA97   18          CLC
 AA98   A5 AA       LDA   $AA       ;$AA,$AB = $AA,$AB + KMX
 AA9A   6D 4B CB    ADC   KMX
 AA9D   85 AA       STA   $AA
 AA9F   A5 AB       LDA   $AB
 AAA1   6D 4C CB    ADC   KMX+1
 AAA4   85 AB       STA   $AB
 AAA6   60          RTS
```

```
KMYPLUS:
 AAA7   18          CLC
 AAA8   A5 AE       LDA   $AE       ;$AE = $AE + KMY
 AAAA   6D 49 CB    ADC   KMY
 AAAD   85 AE       STA   $AE
 AAAF   60          RTS

KMXMINUS:
 AAB0   38          SEC
 AAB1   AD 4B CB    LDA   KMX       ;$AA,$AB = KMX - $AA,$AB
 AAB4   E5 AA       SBC   $AA
 AAB6   85 AA       STA   $AA
 AAB8   AD 4C CB    LDA   KMX+1
 AABB   E5 AB       SBC   $AB
 AABD   85 AB       STA   $AB
 AABF   B0 06       BCS   $AAC7     ;nicht negativ ?
 AAC1   A9 00       LDA   #$00      ;sonst $AA,$AB = 0
 AAC3   85 AA       STA   $AA
 AAC5   85 AB       STA   $AB
 AAC7   60          RTS

KMYMINUS:
 AAC8   38          SEC
 AAC9   AD 49 CB    LDA   KMY       ;$AE = KMY - $AE
 AACC   E5 AE       SBC   $AE
 AACE   85 AE       STA   $AE
 AAD0   B0 F5       BCS   $AAC7     ;nicht negativ ?
 AAD2   A9 00       LDA   #$00      ;sonst $AE = 0
 AAD4   85 AE       STA   $AE
 AAD6   60          RTS

A8A9DIV90:
 AAD7   A9 5A       LDA   #$5A      ;90 als Divisor
 AAD9   8D BB CB    STA   IDIVSOR
 AADC   A9 00       LDA   #$00
 AADE   8D BC CB    STA   IDIVSOR+1
 AAE1   A5 A8       LDA   $A8       ;$A8,$A9 als Dividend
 AAE3   8D C0 CB    STA   IDIVDEND
 AAE6   A5 A9       LDA   $A9
 AAE8   8D C1 CB    STA   IDIVDEND+1
 AAEB   20 45 8F    JSR   IDIV      ;Integer-Division
 AAEE   AD C2 CB    LDA   IDIVREST  ;Rest nach $A8,$A9
 AAF1   85 A8       STA   $A8      ;($A8,$A9 = $A8,$A9 mod 90)
 AAF3   AD C3 CB    LDA   IDIVREST+1
 AAF6   85 A9       STA   $A9
 AAF8   AD C0 CB    LDA   IDIVDEND  ;Div.-Ergebnis nach $20
 AAFB   85 20       STA   $20       ;($20 = $A8,$A9 div 90)
 AAFD   C9 01       CMP   #$01
 AAFF   F0 05       BEQ   $AB06     ;1 ?
 AB01   C9 03       CMP   #$03
 AB03   F0 01       BEQ   $AB06     ;3 ?
```

```
 AB05   60          RTS                  ;sonst fertig

 AB07   A9 5A       LDA   #$5A           ;$A8 = 90 - $A8
 AB09   E5 A8       SBC   $A8
 AB0B   85 A8       STA   $A8
 AB0D   A5 20       LDA   $20            ;Akku und Flags von $20
 AB0F   60          RTS

BEFANGL:
 AB10   20 DB 83    JSR   INCBASBZ  ;Code ueberlesen
 AB13   20 27 94    JSR   SGETADR   ;Mittelp. X holen
 AB16   8C 4B CB    STY   KMX
 AB19   8C AA C5    STY   EX        ;als Endpunkt X merken
 AB1C   8D 4C CB    STA   KMX+1
 AB1F   8D AB C5    STA   EX+1
 AB22   20 31 80    JSR   SCHKCOM   ;Komma ?
 AB25   20 27 94    JSR   SGETADR   ;Mittelp. Y holen
 AB28   8C 49 CB    STY   KMY
 AB2B   8C A8 C5    STY   EY        ;als Endpunkt Y merken
 AB2E   8D 4A CB    STA   KMY+1
 AB31   8D A9 C5    STA   EY+1
 AB34   20 31 80    JSR   SCHKCOM   ;Komma ?
 AB37   20 27 94    JSR   SGETADR   ;Winkel holen
 AB3A   84 A8       STY   $A8       ;nach $A8,$A9
 AB3C   85 A9       STA   $A9
 AB3E   20 FC 81    JSR   SGETBYTC  ;Radius X holen
 AB41   8E 48 CB    STX   KRX
 AB44   20 FC 81    JSR   SGETBYTC  ;Radius Y holen
 AB47   8E C5 CB    STX   KRY
 AB4A   20 7D 93    JSR   GETPKTF   ;Punktfarbe holen
 AB4D   A9 0A       LDA   #$0A      ;Mehrpunkte-Flag
 AB4F   85 21       STA   $21       ;loeschen
 AB51   20 4F A9    JSR   KREIS1    ;allg. Kreisroutine
 AB54   4C 2A 82    JMP   ENDSMB    ;Befehl fertig

 AB57   20 16 A6    JSR   VGLAE     ;Vergleichsroutine
 AB5A   F0 0A       BEQ   $AB66     ;=, dann Schleife
 AB5C   B0 08       BCS   $AB66     ;kleiner, dann Schleife
 AB5E   AD 2F C5    LDA   UEBERDREH ;Ueberlaufflag
 AB61   C9 0A       CMP   #$0A
 AB63   D0 01       BNE   $AB66      ;nicht ges., dann weiter
 AB65   60          RTS

 AB66   4C 4F A9    JMP   KREIS1     ;zur Schleife

 ;Umwandlung einer Programmzeile (Simon's Befehle)

 AB69   C8          INY              ;Zeiger erhoehen
 AB6A   B9 FB 01    LDA   PZEILB,Y   ;naechstes Zeichen
 AB6D   D0 FA       BNE   $AB69      ;kein Endez.,dann Schl.
 AB6F   4C 7B AB    JMP   $AB7B      ;Umwandlung fertig
```

```
 AB72  60          RTS            ;fertig

 AB73  4C 49 AC    JMP  $AC49

 AB76  4C 69 AB    JMP  $AB69

ZEIUMW1:
 AB79  A0 05       LDY  #$05     ;Zeiger auf 5+$01FB = $0200
 AB7B  B9 FB 01    LDA  PZEILB,Y  ;Zeichen aus Programmzeile
 AB7E  D0 01       BNE  $AB81     ;kein Endezeichen ?
 AB80  60          RTS            ;Umwandlung fertig

 AB81  C9 22       CMP  #$22     ;Anfuehrungszeichen ?
 AB83  D0 03       BNE  $AB88    ;nein ?
 AB85  4C 50 AC    JMP  $AC50

 AB88  C9 40       CMP  #$40     ;(Klammeraffe)
 AB8A  F0 EA       BEQ  $AB76    ;ja ?
 AB8C  C9 44       CMP  #$44     ;"E"
 AB8E  D0 24       BNE  $ABB4    ;ja ?
 AB90  48          PHA           ;Akku und Y sichern
 AB91  98          TYA
 AB92  48          PHA
 AB93  C8          INY
 AB94  B9 FB 01    LDA  PZEILB,Y
 AB97  C9 41       CMP  #$41     ;"A"
 AB99  D0 16       BNE  $ABB1    ;nein ?
 AB9B  C8          INY           ;naechstes Zeichen
 AB9C  B9 FB 01    LDA  PZEILB,Y
 AB9F  C9 54       CMP  #$54     ;"T"
 ABA1  D0 0E       BNE  $ABB1    ;nein ?
 ABA3  C8          INY           ;naechstes Zeichen
 ABA4  B9 FB 01    LDA  PZEILB,Y
 ABA7  C9 41       CMP  #$41     ;"A"
 ABA9  D0 06       BNE  $ABB1    ;nein ?
 ABAB  68          PLA           ;Y und Akku restaurieren
 ABAC  A8          TAY
 ABAD  68          PLA
 ABAE  4C 76 AB    JMP  $AB76    ;fertig

 ABB1  68          PLA           ;Y und Akku restaurieren
 ABB2  A8          TAY
 ABB3  68          PLA
 ABB4  C9 41       CMP  #$41     ;"A"
 ABB6  90 BB       BCC  $AB73    ;kleiner als "A" ?
 ABB8  C9 5B       CMP  #$5B     ;$5A = "Z"
 ABBA  B0 B7       BCS  $AB73    ;groesser als "Z" ?
 ABBC  8C 54 CB    STY  ZWISP2   ;Index Y merken
 ABBF  A2 83       LDX  #$83     ;$20,$21 auf $83E2 (TABBEF)
 ABC1  86 21       STX  $21
 ABC3  A2 E2       LDX  #$E2
```

```
ABC5  86 20     STX  $20
ABC7  A2 00     LDX  #$00
ABC9  8E E4 C5  STX  CODEZAE ;Zaehler f. Befehlsnr. = 0
ABCC  C1 20     CMP  ($20,X) ;mit Zeich. aus TABBEF vgl.
ABCE  F0 1E     BEQ  $ABEE   ;gleich ?
ABD0  A1 20     LDA  ($20,X) ;Zeichen aus TABBEF
ABD2  C9 40     CMP  #$40    ;Trennzeichen (Klammeraffe)
ABD4  F0 25     BEQ  $ABFB   ;ja ?
ABD6  A1 20     LDA  ($20,X) ;Zeichen aus TABBEF
ABD8  F0 6F     BEQ  $AC49   ;Tabelle zu Ende ?
ABDA  C9 40     CMP  #$40    ;(Klammeraffe)
ABDC  F0 09     BEQ  $ABE7   ;ja ?
ABDE  E6 20     INC  $20     ;Zeiger in TABBEF erhoehen
ABE0  D0 F4     BNE  $ABD6
ABE2  E6 21     INC  $21
ABE4  4C D6 AB  JMP  $ABD6   ;Schleife

ABE7  EE E4 C5  INC  CODEZAE ;Zaehler erhoehen
ABEA  AC 54 CB  LDY  ZWISP2  ;alten Index wieder holen
ABED  A9 C8     LDA  #$C8    ;(unsinnig)
ABEF  B9 FB 01  LDA  PZEILB,Y ;Zeichen aus Prg.z.
ABF2  E6 20     INC  $20     ;Zeiger in TABBEF erhoehen
ABF4  D0 02     BNE  $ABF8
ABF6  E6 21     INC  $21
ABF8  4C CC AB  JMP  $ABCC   ;naechsten Befehl aus TAB.

ABFB  AE 54 CB  LDX  ZWISP2  ;alten Index nach X
ABFE  A9 64     LDA  #$64    ;Simon's Prefix
AC00  9D FB 01  STA  PZEILB,X ;speichern
AC03  E8        INX
AC04  AD E4 C5  LDA  CODEZAE ;Befehlsnummer
AC07  C9 40     CMP  #$40    ;'DISAPA'
AC09  D0 1E     BNE  $AC29   ;nein ?
AC0B  9D FB 01  STA  PZEILB,X ;Befehlscode speichern
AC0E  FE FB 01  INC  PZEILB,X ;Befehlscode + 1
AC11  A9 3A     LDA  #$3A     ;Doppelpunkt
AC13  E8        INX
AC14  9D FB 01  STA  PZEILB,X ;4 Doppelpunkte einfuegen
AC17  E8        INX
AC18  9D FB 01  STA  PZEILB,X
AC1B  E8        INX
AC1C  9D FB 01  STA  PZEILB,X
AC1F  E8        INX
AC20  9D FB 01  STA  PZEILB,X
AC23  E8        INX
AC24  8A        TXA
AC25  A8        TAY           ;Index wieder nach Y
AC26  4C 7B AB  JMP  $AB7B    ;naechsten Befehl

AC29  9D FB 01  STA  PZEILB,X ;Befehlscode speichern
AC2C  FE FB 01  INC  PZEILB,X ;+1
```

```
AC2F  88        DEY                  ;Y zeigt auf noch nicht
AC30  C8        INY                  ;kodierte Zeichen
AC31  E8        INX                  ;X auf Umwandlungsergebnis
AC32  B9 FB 01  LDA  PZEILB,Y        ;Byte
AC35  9D FB 01  STA  PZEILB,X        ;nach vorne holen
AC38  D0 F6     BNE  $AC30           ;k. Zeilenende, dann Schl.
AC3A  E8        INX                  ;Zeiger dahinter stellen
AC3B  A9 00     LDA  #$00
AC3D  9D FB 01  STA  PZEILB,X        ;Endemarke setzen
AC40  BD FC 01  LDA  PZEILB+1,X ;naechstes Zeichen
AC43  D0 F5     BNE  $AC3A           ;kein (altes) Zeilenende ?
AC45  AC 54 CB  LDY  ZWISP2          ;alten Index holen
AC48  C8        INY                  ;+2 (Prefix und Code)
AC49  C8        INY
AC4A  4C 7B AB  JMP  $AB7B           ;zum naechsten Befehl

AC4D  4C 72 AB  JMP  $AB72           ;fertig

AC50  C8        INY                  ;naechstes Zeichen
AC51  B9 FB 01  LDA  PZEILB,Y
AC54  F0 F7     BEQ  $AC4D           ;Zeilenende ?
AC56  C9 22     CMP  #$22            ;"
AC58  D0 F6     BNE  $AC50           ;kein ", dann Schleife
AC5A  4C 49 AC  JMP  $AC49           ;weiter

MOVEBEF:    ;gemeinsame Adresse von INV,MOVE,UP,DOWM etc.
AC5D  A9 FF     LDA  #$FF            ;Hilfszaehler auf $FF
AC5F  85 A6     STA  $A6             ;setzen
AC61  A5 7A     LDA  $7A             ;Prg.z. nach $A8,$A9
AC63  85 A8     STA  $A8
AC65  A5 7B     LDA  $7B
AC67  85 A9     STA  $A9
AC69  E6 A6     INC  $A6             ;Hilfszaehler erhoehen
AC6B  A5 A6     LDA  $A6
AC6D  C9 02     CMP  #$02
AC6F  D0 03     BNE  $AC74           ;nicht 2 ?
AC71  4C 2A 82  JMP  ENDSMB          ;Befehl fertig

AC74  A5 A8     LDA  $A8             ;alten Prgogrammzaehler
AC76  85 7A     STA  $7A
AC78  A5 A9     LDA  $A9
AC7A  85 7B     STA  $7B
AC7C  18        CLC
AC7D  A0 00     LDY  #$00
AC7F  B1 7A     LDA  ($7A),Y         ;Code
AC81  69 01     ADC  #$01            ;+1
AC83  4C 0A AD  JMP  $AD0A           ;zum Verteiler

AC86  68        PLA                  ;Stack bereinigen
AC87  68        PLA
AC88  A9 0C     LDA  #$0C            ;Nr. f. 'bad mode'
```

```
 AC8A  4C 8C 88   JMP  SERROUT  ;Fehler ausgeben

GETRCWD:   ;4 Parameter R,C,W,D ;Bez. s. Handbuch bei FCHR
 AC8D  20 02 82   JSR  SGETBYTN ;R holen
 AC90  E0 29      CPX  #$29
 AC92  B0 F2      BCS  $AC86    ;groesser 40 ?
 AC94  8E 76 CB   STX  ZEILEANF
 AC97  20 FC 81   JSR  SGETBYTC ;C holen
 AC9A  E0 28      CPX  #$28
 AC9C  B0 E8      BCS  $AC86    ;groesser 39 ?
 AC9E  8E 77 CB   STX  SPALTEANF
 ACA1  20 FC 81   JSR  SGETBYTC ;W holen
 ACA4  E0 00      CPX  #$00
 ACA6  F0 DE      BEQ  $AC86    ;W = 0 ?
 ACA8  8E 78 CB   STX  SPALTENANZ
 ACAB  20 FC 81   JSR  SGETBYTC ;D holen
 ACAE  E0 00      CPX  #$00
 ACB0  F0 D4      BEQ  $AC86    ;D = 0 ?
 ACB2  8E 79 CB   STX  ZEILENANZ

MOVEZ:                     ;Zeiger f. Move-Bef. initialisieren
 ACB5  A9 00      LDA  #$00
 ACB7  8D 7A CB   STA  ZAEHLER  ;Zaehler auf 0 stellen
 ACBA  85 20      STA  $20      ;$20,$21 auf Video-RAM
 ACBC  AD 88 02   LDA  VIDRAMHI
 ACBF  85 21      STA  $21
 ACC1  A5 A6      LDA  $A6      ;Hilfszaehler
 ACC3  F0 04      BEQ  $ACC9    ;noch 0 ?
 ACC5  A9 D8      LDA  #$D8     ;$20,$21 auf Farb-RAM
 ACC7  85 21      STA  $21
 ACC9  AD 76 CB   LDA  ZEILEANF
 ACCC  85 A4      STA  $A4
 ACCE  A9 00      LDA  #$00
 ACD0  85 A5      STA  $A5
 ACD2  20 5F 94   JSR  A4MAL40  ;$A4,$A5 = 40 * ZEILEANF
 ACD5  18         CLC
 ACD6  A5 A4      LDA  $A4    ;$A4,$A5 = $A4,$A4 + $20,$21
 ACD8  65 20      ADC  $20
 ACDA  85 A4      STA  $A4
 ACDC  A5 A5      LDA  $A5
 ACDE  65 21      ADC  $21
 ACE0  85 A5      STA  $A5
 ACE2  18         CLC
 ACE3  A5 A4      LDA  $A4  ;$20,$21 = $A4,$A5 + SPALTEANF
 ACE5  6D 77 CB   ADC  SPALTEANF
 ACE8  85 20      STA  $20
 ACEA  A5 A5      LDA  $A5
 ACEC  69 00      ADC  #$00
 ACEE  85 21      STA  $21
 ACF0  18         CLC
 ACF1  AD 76 CB   LDA  ZEILEANF
```

```
 ACF4   6D 79 CB    ADC   ZEILENANZ  ;ZEILEANF+ZEILENANZ
 ACF7   C9 1A       CMP   #$1A
 ACF9   B0 0C       BCS   $AD07      ;Zeilen groesser 25 ?
 ACFB   18          CLC
 ACFC   AD 78 CB    LDA   SPALTENANZ
 ACFF   6D 77 CB    ADC   SPALTEANF ;SPALTENANZ+SPALTEANF
 AD02   C9 29       CMP   #$29
 AD04   B0 01       BCS   $AD07      ;Spalten groesser 40 ?
 AD06   60          RTS              ;fertig

 AD07   4C 86 AC    JMP   $AC86      ;zur Fehlermeldung

;Verteiler auf die einzelnen Befehle
 AD0A   C9 0E       CMP   #$0E       ;'INV'
 AD0C   D0 27       BNE   $AD35      ;nein ?

;INV-Befehl
 AD0E   20 8D AC    JSR   GETRCWD    ;Parameter holen
 AD11   A0 00       LDY   #$00       ;Spaltenzaehler
 AD13   A2 00       LDX   #$00       ;Zeilenzaehler
 AD15   CC 78 CB    CPY   SPALTENANZ
 AD18   F0 0A       BEQ   $AD24      ;Ende (Spalten) ?
 AD1A   B1 20       LDA   ($20),Y
 AD1C   49 80       EOR   #$80       ;MSB invertieren
 AD1E   91 20       STA   ($20),Y
 AD20   C8          INY
 AD21   4C 15 AD    JMP   $AD15      ;Schleife

 AD24   E8          INX
 AD25   EC 79 CB    CPX   ZEILENANZ
 AD28   F0 08       BEQ   $AD32      ;Ende ?
 AD2A   A0 00       LDY   #$00
 AD2C   20 CE 94    JSR   20PL40     ;naechste Zeile
 AD2F   4C 15 AD    JMP   $AD15      ;Schleife

 AD32   4C 2A 82    JMP   ENDSMB     ;Befehl fertig

;Fortsetzung des Verteilers
 AD35   C9 10       CMP   #$10       ;'MOVE'
 AD37   F0 03       BEQ   $AD3C      ;ja ?
 AD39   4C 7B AD    JMP   $AD7B

;MOVE-Befehl
 AD3C   20 8D AC    JSR   GETRCWD    ;4 Parameter holen
 AD3F   A5 20       LDA   $20        ;$23,$24 = $20,$21
 AD41   85 23       STA   $23
 AD43   A5 21       LDA   $21
 AD45   85 24       STA   $24
 AD47   20 FC 81    JSR   SGETBYTC ;Zielzeile holen
 AD4A   8E 76 CB    STX   ZEILEANF
 AD4D   20 FC 81    JSR   SGETBYTC ;Zielspalte holen
```

```
 AD50   8E 77 CB    STX   SPALTEANF
 AD53   20 B5 AC    JSR   MOVEZ       ;Zeiger initialisieren
 AD56   A2 00       LDX   #$00        ;Zeilenzaehler
 AD58   A0 00       LDY   #$00        ;Spaltenzaehler
 AD5A   CC 78 CB    CPY   SPALTENANZ
 AD5D   F0 08       BEQ   $AD67       ;Spalten fertig ?
 AD5F   B1 23       LDA   ($23),Y     ;von  ($23,$24 + Y)
 AD61   91 20       STA   ($20),Y     ;nach ($20,$21 + Y)
 AD63   C8          INY
 AD64   4C 5A AD    JMP   $AD5A       ;Schleife

 AD67   E8          INX
 AD68   EC 79 CB    CPX   ZEILENANZ
 AD6B   F0 0B       BEQ   $AD78       ;Zeilen fertig ?
 AD6D   A0 00       LDY   #$00
 AD6F   20 CE 94    JSR   20PL40      ;Quellzeiger und Zielzeig.
 AD72   20 DC 94    JSR   23PL40      ;auf naechste Zeile
 AD75   4C 5A AD    JMP   $AD5A       ;Schleife

 AD78   4C 69 AC    JMP   $AC69       ;mit Speicherber. fertig

;Fortsetzung des Verteilers
 AD7B   C9 12       CMP   #$12        ;'UPB'
 AD7D   F0 03       BEQ   $AD82       ;ja ?
 AD7F   4C 1F AE    JMP   $AE1F

;UPB-Befehl
 AD82   A9 28       LDA   #$28
 AD84   8D 7C CB    STA   RICHTZ      ;up
 AD87   20 8D AD    JSR   MOVEUL      ;Move nach oben bzw. links
 AD8A   4C 0F AE    JMP   $AE0F       ;Abschluss

MOVEUL:
 AD8D   20 8D AC    JSR   GETRCWD     ;4 Parameter holen
 AD90   18          CLC
 AD91   A5 20       LDA   $20        ;$23,$24 = $20,$21 + RICHTZ
 AD93   6D 7C CB    ADC   RICHTZ
 AD96   85 23       STA   $23
 AD98   A5 21       LDA   $21
 AD9A   69 00       ADC   #$00
 AD9C   85 24       STA   $24
 AD9E   A2 00       LDX   #$00
 ADA0   A0 00       LDY   #$00
 ADA2   AD 7C CB    LDA   RICHTZ
 ADA5   C9 28       CMP   #$28
 ADA7   F0 30       BEQ   $ADD9      ;RICHTZ = 40 ?
 ADA9   CE 78 CB    DEC   SPALTENANZ   ;-1, aber nicht null
 ADAC   AD 78 CB    LDA   SPALTENANZ
 ADAF   D0 03       BNE   $ADB4
 ADB1   EE 78 CB    INC   SPALTENANZ
 ADB4   A5 20       LDA   $20
```

```
ADB6  8D 7E CB   STA  ZAEHLM1  ;$20,$21 nach ZAEHLM1,2
ADB9  A5 21      LDA  $21
ADBB  8D 80 CB   STA  ZAEHLM2
ADBE  B1 20      LDA  ($20),Y  ;eine Spalte nach MOVETAB
ADC0  9D 1E CB   STA  MOVETAB,X
ADC3  E8         INX
ADC4  20 CE 94   JSR  20PL40
ADC7  EC 79 CB   CPX  ZEILENANZ
ADCA  D0 F2      BNE  $ADBE
ADCC  AD 80 CB   LDA  ZAEHLM2   ;$20,$21 restaurieren
ADCF  85 21      STA  $21
ADD1  AD 7E CB   LDA  ZAEHLM1
ADD4  85 20      STA  $20
ADD6  4C E7 AD   JMP  $ADE7

ADD9  B1 20      LDA  ($20),Y   ;eine Zeile nach MOVETAB
ADDB  99 1E CB   STA  MOVETAB,Y
ADDE  C8         INY
ADDF  CC 78 CB   CPY  SPALTENANZ
ADE2  D0 F5      BNE  $ADD9
ADE4  CE 79 CB   DEC  ZEILENANZ
ADE7  A0 00      LDY  #$00      ;Bereich moven
ADE9  A2 00      LDX  #$00
ADEB  CC 78 CB   CPY  SPALTENANZ
ADEE  F0 08      BEQ  $ADF8
ADF0  B1 23      LDA  ($23),Y
ADF2  91 20      STA  ($20),Y
ADF4  C8         INY
ADF5  4C EB AD   JMP  $ADEB

ADF8  E8         INX
ADF9  EC 79 CB   CPX  ZEILENANZ
ADFC  F0 10      BEQ  $AE0E
ADFE  AD 79 CB   LDA  ZEILENANZ
AE01  F0 0B      BEQ  $AE0E
AE03  A0 00      LDY  #$00
AE05  20 CE 94   JSR  20PL40     ;Zielzeiger n. Zeile
AE08  20 DC 94   JSR  23PL40     ;Quellzeiger
AE0B  4C EB AD   JMP  $ADEB

AE0E  60         RTS             ;fertig mit MOVEUL

AE0F  AC 78 CB   LDY  SPALTENANZ
AE12  88         DEY
AE13  A9 20      LDA  #$20   ;Zeile mit Blanks auffuellen
AE15  91 23      STA  ($23),Y
AE17  88         DEY
AE18  C0 FF      CPY  #$FF
AE1A  D0 F9      BNE  $AE15  ;Schleife
AE1C  4C 69 AC   JMP  $AC69  ;mit Speicherber. fertig
```

```
;Fortsetzung des Verteilers
 AE1F   C9 13        CMP  #$13      ;'UPW'
 AE21   F0 03        BEQ  $AE26     ;ja ?
 AE23   4C 3F AE     JMP  $AE3F

UPW-Befehl
 AE26   A9 28        LDA  #$28
 AE28   8D 7C CB     STA  RICHTZ    ;up
 AE2B   20 8D AD     JSR  MOVEUL    ;Move nach oben bzw. links
 AE2E   AC 78 CB     LDY  SPALTENANZ
 AE31   88           DEY
 AE32   B9 1E CB     LDA  MOVETAB,Y ;gesicherte Zeile aus
 AE35   91 23        STA  ($23),Y    ;wieder schreiben
 AE37   88           DEY
 AE38   C0 FF        CPY  #$FF
 AE3A   D0 F6        BNE  $AE32      ;Schleife
 AE3C   4C 69 AC     JMP  $AC69      ;mit Bereich fertig

;Fortsetzung des Verteilers
 AE3F   C9 14        CMP  #$14      ;'LEFTW'
 AE41   F0 03        BEQ  $AE46     ;ja ?
 AE43   4C 98 AE     JMP  $AE98

;LEFTW-Befehl
 AE46   A9 01        LDA  #$01
 AE48   8D 7C CB     STA  RICHTZ    ;left
 AE4B   8D 82 CB     STA  BWFLAG
 AE4E   20 8D AD     JSR  MOVEUL    ;Move nach oben bzw. links
 AE51   A2 00        LDX  #$00
 AE53   AC 79 CB     LDY  ZEILENANZ
 AE56   C0 01        CPY  #$01
 AE58   F0 12        BEQ  $AE6C
 AE5A   18           CLC
 AE5B   A5 23        LDA  $23       ;$23,$24 = $23,$24 - 40
 AE5D   E9 27        SBC  #$27
 AE5F   85 23        STA  $23
 AE61   A5 24        LDA  $24
 AE63   E9 00        SBC  #$00
 AE65   85 24        STA  $24
 AE67   88           DEY
 AE68   C0 01        CPY  #$01
 AE6A   D0 EE        BNE  $AE5A     ;Schleife f. Subtr.
 AE6C   AC 78 CB     LDY  SPALTENANZ
 AE6F   88           DEY
 AE70   AD 82 CB     LDA  BWFLAG    ;LEFTB oder LEFTW ?
 AE73   C9 04        CMP  #$04
 AE75   F0 11        BEQ  $AE88     ;LEFTB ?
 AE77   BD 1E CB     LDA  MOVETAB,X ;gesicherte Spalte
 AE7A   91 23        STA  ($23),Y    ;wieder schreiben
 AE7C   20 DC 94     JSR  23PL40     ;naechste Zeile
 AE7F   E8           INX
```

```
 AE80   EC 79 CB    CPX   ZEILENANZ
 AE83   D0 F2       BNE   $AE77       ;Schleife
 AE85   4C 69 AC    JMP   $AC69       ;mit Ber. fertig

 AE88   A9 20       LDA   #$20        ;" "
 AE8A   91 23       STA   ($23),Y     ;in Spalte schr.
 AE8C   20 DC 94    JSR   23PL40
 AE8F   E8          INX
 AE90   EC 79 CB    CPX   ZEILENANZ
 AE93   D0 F3       BNE   $AE88       ;Schleife
 AE95   4C 69 AC    JMP   $AC69       ;mit Bereich fertig

;Fortsetzung des Verteilers
 AE98   C9 15       CMP   #$15        ;'LEFTB'
 AE9A   F0 03       BEQ   $AE9F       ;ja ?
 AE9C   4C AF AE    JMP   $AEAF

;LEFTB-Befehl
 AE9F   A9 01       LDA   #$01
 AEA1   8D 7C CB    STA   RICHTZ      ;left
 AEA4   A9 04       LDA   #$04
 AEA6   8D 82 CB    STA   BWFLAG      ;left B
 AEA9   20 8D AD    JSR   MOVEUL      ;Move-Routine
 AEAC   4C 51 AE    JMP   $AE51       ;Abschlussbehandlung

;Fortsetzung des Verteilers
 AEAF   C9 16       CMP   #$16        ;'DOWNMB'
 AEB1   F0 03       BEQ   $AEB6       ;ja ?
 AEB3   4C 79 AF    JMP   $AF79

;DOWNB-Befehl
 AEB6   A9 28       LDA   #$28
 AEB8   8D 7C CB    STA   RICHTZ      ;down
 AEBB   A9 00       LDA   #$00
 AEBD   8D 7E CB    STA   ZAEHLM1
 AEC0   20 C6 AE    JSR   MOVEDR      ;Move n. unten bzw. rechts
 AEC3   4C 6A AF    JMP   $AF6A       ;Abschlussbehandlung

MOVEDR:
 AEC6   20 8D AC    JSR   GETRCWD  ;4 Parameter holen
 AEC9   CE 79 CB    DEC   ZEILENANZ
 AECC   CE 78 CB    DEC   SPALTENANZ
 AECF   AE 79 CB    LDX   ZEILENANZ
 AED2   A0 00       LDY   #$00
 AED4   18          CLC
 AED5   8C 84 CB    STY   ZWISP3     ;Y sichern
 AED8   AC 78 CB    LDY   SPALTENANZ
 AEDB   B1 20       LDA   ($20),Y    ;Zeichen
 AEDD   AC 84 CB    LDY   ZWISP3
 AEE0   99 1E CB    STA   MOVETAB,Y ;in MOVETAB sichern
 AEE3   AD 79 CB    LDA   ZEILENANZ
```

```
AEE6  F0 0A     BEQ  $AEF2     ;fertig mit Zeilen
AEE8  C8        INY
AEE9  20 CE 94  JSR  20PL40    ;naechste Zeile
AEEC  8C 84 CB  STY  ZWISP3    ;Y sichern
AEEF  CA        DEX
AEF0  D0 E2     BNE  $AED4     ;Schleife
AEF2  8C 84 CB  STY  ZWISP3
AEF5  AC 78 CB  LDY  SPALTENANZ
AEF8  B1 20     LDA  ($20),Y   ;Spalte
AEFA  AC 84 CB  LDY  ZWISP3
AEFD  99 1E CB  STA  MOVETAB,Y ;in MOVETAB sichern
AF00  38        SEC
AF01  A5 20     LDA  $20      ;$23,$24 = $20,$21 - RICHTZ
AF03  ED 7C CB  SBC  RICHTZ
AF06  85 23     STA  $23
AF08  A5 21     LDA  $21
AF0A  E9 00     SBC  #$00
AF0C  85 24     STA  $24
AF0E  AD 7E CB  LDA  ZAEHLM1
AF11  C9 28     CMP  #$28
AF13  D0 1D     BNE  $AF32     ;ZAEHLM1 nicht= 40 ?
AF15  A0 00     LDY  #$00
AF17  EE 78 CB  INC  SPALTENANZ
AF1A  AD 80 CB  LDA  ZAEHLM2
AF1D  C9 02     CMP  #$02
AF1F  F0 0E     BEQ  $AF2F
AF21  B1 20     LDA  ($20),Y    ;Zeile sichern in MOVETAB
AF23  99 1E CB  STA  MOVETAB,Y
AF26  C8        INY
AF27  CC 78 CB  CPY  SPALTENANZ
AF2A  F0 03     BEQ  $AF2F      ;mit Zeile fertig ?
AF2C  4C 21 AF  JMP  $AF21      ;Schleife

AF2F  CE 78 CB  DEC  SPALTENANZ
AF32  AC 78 CB  LDY  SPALTENANZ
AF35  AE 79 CB  LDX  ZEILENANZ
AF38  C0 FF     CPY  #$FF
AF3A  F0 08     BEQ  $AF44
AF3C  B1 23     LDA  ($23),Y   ;Zeichen transferieren
AF3E  91 20     STA  ($20),Y
AF40  88        DEY
AF41  4C 38 AF  JMP  $AF38     ;Schleife

AF44  CA        DEX
AF45  E0 FF     CPX  #$FF
AF47  F0 20     BEQ  $AF69      ;Zeilenz. abgelaufen ?
AF49  AC 78 CB  LDY  SPALTENANZ
AF4C  38        SEC
AF4D  A5 20     LDA  $20     ;$20,$21 = $20,$21 - 40
AF4F  E9 28     SBC  #$28
AF51  85 20     STA  $20
```

```
AF53   A5 21       LDA  $21
AF55   E9 00       SBC  #$00
AF57   85 21       STA  $21
AF59   38          SEC
AF5A   A5 23       LDA  $23         $23,$24 = $23,$24 - 40
AF5C   E9 28       SBC  #$28
AF5E   85 23       STA  $23
AF60   A5 24       LDA  $24
AF62   E9 00       SBC  #$00
AF64   85 24       STA  $24
AF66   4C 38 AF    JMP  $AF38       ;Schleife

AF69   60          RTS              ;Ende von MOVEDR

AF6A   AC 78 CB    LDY  SPALTENANZ
AF6D   A9 20       LDA  #$20        ;" "
AF6F   91 20       STA  ($20),Y     ;in neue Zeile schreiben
AF71   88          DEY
AF72   C0 FF       CPY  #$FF
AF74   D0 F9       BNE  $AF6F       ;Schleife
AF76   4C 69 AC    JMP  $AC69       ;mit Bereich fertig

;Fortsetzung des Verteilers
AF79   C9 17       CMP  #$17        ;'DOWNW'
AF7B   F0 03       BEQ  $AF80       ;ja ?
AF7D   4C A4 AF    JMP  $AFA4

;DOWNW-Befehl
AF80   A9 28       LDA  #$28        ;down
AF82   8D 7C CB    STA  RICHTZ
AF85   8D 7E CB    STA  ZAEHLM1
AF88   8D 80 CB    STA  ZAEHLM2
AF8B   20 C6 AE    JSR  MOVEDR      ;move n. unten oder rechts
AF8E   EE 78 CB    INC  SPALTENANZ
AF91   A0 00       LDY  #$00
AF93   B9 1E CB    LDA  MOVETAB,Y   ;gesicherte Zeile
AF96   91 20       STA  ($20),Y     ;wieder schreiben
AF98   C8          INY
AF99   CC 78 CB    CPY  SPALTENANZ
AF9C   10 03       BPL  $AFA1       ;mit Spalten fertig ?
AF9E   4C 93 AF    JMP  $AF93       ;Schleife

AFA1   4C 69 AC    JMP  $AC69       ;Bereich fertig

;Fortsetzung des Verteilers
AFA4   C9 18       CMP  #$18        ;'RIGHTB'
AFA6   F0 03       BEQ  $AFAB       ;ja ?
AFA8   4C D0 AF    JMP  $AFD0

;RIGHTB-Befehl
AFAB   A9 01       LDA  #$01        ;right
```

```
 AFAD   8D 7C CB   STA   RICHTZ
 AFB0   8D 7E CB   STA   ZAEHLM1
 AFB3   20 C6 AE   JSR   MOVEDR     ;move n. unten oder rechts
 AFB6   A2 00      LDX   #$00
 AFB8   EE 79 CB   INC   ZEILENANZ
 AFBB   A0 00      LDY   #$00
 AFBD   A9 20      LDA   #$20       ;" "
 AFBF   91 20      STA   ($20),Y    ;in neue Spalte
 AFC1   E8         INX
 AFC2   20 CE 94   JSR   20PL40     ;naechste Spalte
 AFC5   EC 79 CB   CPX   ZEILENANZ
 AFC8   10 03      BPL   $AFCD      ;mit Zeilen fertig ?
 AFCA   4C BD AF   JMP   $AFBD      ;Schleife

 AFCD   4C 69 AC   JMP   $AC69      ;Bereich fertig

;Fortsetzung des Verteilers
 AFD0   C9 19      CMP   #$19       ;'RIGHTW'
 AFD2   F0 00      BEQ   $AFD4      ;ja ?
 AFD4   A9 01      LDA   #$01       ;right
 AFD6   8D 7C CB   STA   RICHTZ
 AFD9   8D 7E CB   STA   ZAEHLM1
 AFDC   A9 02      LDA   #$02
 AFDE   8D 80 CB   STA   ZAEHLM2
 AFE1   20 C6 AE   JSR   MOVEDR     ;move n. unten oder rechts
 AFE4   A0 00      LDY   #$00
 AFE6   A2 00      LDX   #$00
 AFE8   EE 79 CB   INC   ZEILENANZ
 AFEB   BD 1E CB   LDA   MOVETAB,X ;gesicherte Spalte
 AFEE   91 20      STA   ($20),Y    ;wieder schreiben
 AFF0   E8         INX
 AFF1   20 CE 94   JSR   20PL40     ;naechste Spalte
 AFF4   EC 79 CB   CPX   ZEILENANZ
 AFF7   F0 03      BEQ   $AFFC      ;mit Zeilen fertig ?
 AFF9   4C EB AF   JMP   $AFEB      ;Schleife

 AFFC   A9 01      LDA   #$01
 AFFE   8D 84 CB   STA   ZWISP3     ;Y auf 1 setzen
 B001   4C 69 AC   JMP   $AC69      ;Bereich fertig

BEFFETCH:
 B004   20 B4 86   JSR   SGETSTRN   ;String holen
 B007   86 20      STX   $20        ;Adresse nach $20,$21
 B009   85 21      STA   $21
 B00B   A5 69      LDA   $69
 B00D   8D 72 CB   STA   FETCHLEN   ;Stringlaenge
 B010   20 FC 81   JSR   SGETBYTC   ;Anzahl holen
 B013   8E 6F CB   STX   FETCHANZ
 B016   20 31 80   JSR   SCHKCOM    ;Komma ?
 B019   20 70 B1   JSR   RVSBLANK   ;RVS-Blank ausgeben
 B01C   20 C7 80   JSR   STESTDIRM ;Programmodus ?
```

```
B01F  A9 2C     LDA  #$2C       ;","
B021  8D FF 01  STA  $01FF      ;an Anfang von Puffer
B024  20 53 B0  JSR  FETCHT     ;Zeichen holen
B027  A5 13     LDA  $13        ;aktive Datei
B029  F0 10     BEQ  $B03B      ;0, dann weiter
B02B  20 B7 FF  JSR  $FFB7      ;READST; Status holen
B02E  29 02     AND  #$02       ;Time-Out read
B030  F0 09     BEQ  $B03B      ;nein, dann weiter
B032  20 D0 80  JSR  SCLRCH     ;Kanaele schliessen
B035  20 D9 80  JSR  SREM       ;bis Zeilenende vorr.
B038  4C 2A 82  JMP  ENDSMB     ;Befehl fertig

B03B  AD 00 02  LDA  $0200      ;1. Zeich. aus Eing.puffer
B03E  D0 0D     BNE  $B04D      ;kein Endezeichen ?
B040  A5 13     LDA  $13        ;aktive Datei
B042  D0 E0     BNE  $B024      ;nicht 0, dann Schleife
B044  20 E2 80  JSR  SNEXTTR    ;naechsten Befehl suchen
B047  20 FD 80  JSR  SBZPLY     ;Prg.z. erhoehen
B04A  4C 2A 82  JMP  ENDSMB     ;Befehl fertig

B04D  20 EB 80  JSR  SINPUT1    ;Input-Routine
B050  4C 2A 82  JMP  ENDSMB     ;Befehl fertig

FETCHT:
B053  A2 00     LDX  #$00
B055  98        TYA             ;Y und X retten
B056  48        PHA
B057  8A        TXA
B058  48        PHA
B059  20 E4 FF  JSR  GETIN      ;Zeichen holen
B05C  8D 70 CB  STA  JOYWERT    ;merken
B05F  68        PLA
B060  AA        TAX             ;X und Y restaurieren
B061  68        PLA
B062  A8        TAY
B063  AD 70 CB  LDA  JOYWERT    ;geholtes Zeichen
B066  C9 00     CMP  #$00
B068  F0 EB     BEQ  $B055      ;kein Zeichen ?
B06A  C9 0D     CMP  #$0D
B06C  D0 03     BNE  $B071      ;kein (return) ?
B06E  4C 0D B1  JMP  $B10D      ;weiter
B071  C9 14     CMP  #$14
B073  F0 22     BEQ  $B097      ;(Del) ?
B075  A5 D4     LDA  $D4        ;"-Merker
B077  D0 36     BNE  $B0AF      ;kein "-Modus ?
B079  8A        TXA             ;X retten
B07A  48        PHA
B07B  A2 00     LDX  #$00       ;Zaehler auf 0
B07D  BD 5B B1  LDA  FETCHTAB,X ;Zeichen aus Tabelle
B080  CD 70 CB  CMP  JOYWERT    ;mit Zeichen vgl.
B083  F0 0D     BEQ  $B092      ;gleich ?
```

```
B085  E8         INX                ;naechtes Zeichen in Tab.
B086  E0 12      CPX   #$12
B088  D0 F3      BNE   $B07D        ;Tabelle nicht zu Ende ?
B08A  68         PLA
B08B  AA         TAX                ;X restaurieren
B08C  AD 70 CB   LDA   JOYWERT      ;geholtes Zeichen
B08F  4C AF B0   JMP   $B0AF

B092  68         PLA
B093  AA         TAX                ;X restaurieren
B094  4C 55 B0   JMP   $B055        ;neues Zeichen holen

B097  E0 00      CPX   #$00
B099  F0 BA      BEQ   $B055        ;Zaehler = 0 ?
B09B  A5 D4      LDA   $D4          ;"-Merker retten
B09D  48         PHA
B09E  A9 00      LDA   #$00
B0A0  85 D4      STA   $D4
B0A2  A9 14      LDA   #$14         ;(Del)
B0A4  20 D2 FF   JSR   BSOUT        ;ausgeben
B0A7  68         PLA
B0A8  85 D4      STA   $D4          ;"-Merker restaurieren
B0AA  CA         DEX              ;2 Zeichen aus Puffer nehmen
B0AB  CA         DEX
B0AC  4C EF B0   JMP   $B0EF      ;weiter

B0AF  8C 71 CB   STY   ZWISP4     ;Y sichern
B0B2  AD 70 CB   LDA   JOYWERT    ;Taste
B0B5  C9 8D      CMP   #$8D       ;(Shift+RETURN)
B0B7  F0 E0      BEQ   $B099      ;ja ?
B0B9  AC 73 CB   LDY   $CB73        ;????? (immer 0)
B0BC  C0 78      CPY   #$78
B0BE  F0 26      BEQ   $B0E6        ;????? (nie erfuellt)
B0C0  A0 00      LDY   #$00
B0C2  D1 20      CMP   ($20),Y      ;Zeichen aus String
B0C4  F0 20      BEQ   $B0E6        ;gleich Eingabezeichen ?
B0C6  B1 20      LDA   ($20),Y      ;Zeichen aus String
B0C8  C9 13      CMP   #$13
B0CA  F0 57      BEQ   $B123        ;chr$(13) ?
B0CC  C9 11      CMP   #$11
B0CE  F0 64      BEQ   $B134        ;(Cursor down) ?
B0D0  C9 1D      CMP   #$1D
B0D2  D0 03      BNE   $B0D7        ;nicht (Cursor right) ?
B0D4  4C 42 B1   JMP   $B142        ;ja, dann zu $B142

B0D7  AD 70 CB   LDA   JOYWERT      ;gedrueckte Taste
B0DA  C8         INY                ;Zaehler im String erhoeh.
B0DB  CC 72 CB   CPY   FETCHLEN     ;Stringlaenge
B0DE  D0 E2      BNE   $B0C2        ;String nicht zu Ende ?
B0E0  AC 71 CB   LDY   ZWISP4       ;Y restaurieren
B0E3  4C 55 B0   JMP   $B055        ;zur Schleife
```

```
B0E6   9D 00 02   STA  $0200,X    ;Zeichen in Puffer ablegen
B0E9   AC 71 CB   LDY  ZWISP4     ;Y restaurieren
B0EC   20 D2 FF   JSR  BSOUT      ;Zeichen ausgeben
B0EF   20 70 B1   JSR  RVSBLANK   ;RVS-Blank ausgeben
B0F2   E8         INX             ;Zaehler erhoehen
B0F3   EC 6F CB   CPX  FETCHANZ   ;zu holende Zeichenzahl
B0F6   D0 E8      BNE  $B0E0      ;nicht erreicht ?
B0F8   98         TYA             ;Y und X sichern
B0F9   48         PHA
B0FA   8A         TXA
B0FB   48         PHA
B0FC   20 E4 FF   JSR  GETIN      ;1 Zeichen holen
B0FF   8D 70 CB   STA  JOYWERT    ;speichern
B102   68         PLA
B103   AA         TAX             ;X und Y restaurieren
B104   68         PLA
B105   A8         TAY
B106   AD 70 CB   LDA  JOYWERT    ;Zeichen
B109   C9 0D      CMP  #$0D       ;(cr)
B10B   D0 EB      BNE  $B0F8      ;nein ?
B10D   E0 00      CPX  #$00
B10F   D0 03      BNE  $B114      ;Zaehler ungleich 0 ?
B111   4C 53 B0   JMP  $B053      ;sonst Schleife

B114   A9 20      LDA  #$20       ;" "
B116   20 D2 FF   JSR  BSOUT      ;ausgeben
B119   A9 91      LDA  #$91       ;(cursor up)
B11B   20 D2 FF   JSR  BSOUT      ;ausgeben
B11E   A9 0D      LDA  #$0D       ;(cr)
B120   4C F4 80   JMP  SPUFOCR    ;Puffer abschliessen;
                                  ;(cr) ausgeben und fertig

B123   AD 70 CB   LDA  JOYWERT    ;geholtes Zeichen
B126   C9 41      CMP  #$41       ;"A"
B128   90 07      BCC  $B131      ;kleiner als "A" ?
B12A   C9 5B      CMP  #$5B       ;$5A = "Z"
B12C   B0 03      BCS  $B131      ;groesser als "Z" ?
B12E   4C E6 B0   JMP  $B0E6      ;Buchstabe ?

B131   4C D7 B0   JMP  $B0D7      ;s.o.

B134   AD 70 CB   LDA  JOYWERT    ;geholtes Zeichen
B137   C9 1F      CMP  #$1F       ;$20 = " "
B139   90 F6      BCC  $B131      ;Steuerzeichen ?
B13B   C9 41      CMP  #$41       ;"A"
B13D   B0 F2      BCS  $B131      ;groesser als "A" ?
B13F   4C E6 B0   JMP  $B0E6      ;" " bis "(Klammeraffe)" ?

B142   AD 70 CB   LDA  JOYWERT    ;geholtes Zeichen
B145   C9 41      CMP  #$41       ;"A"
B147   90 E8      BCC  $B131      ;kleiner als "A" ?
```

```
 B149   C9 5B        CMP   #$5B       ;$5A = "Z"
 B14B   B0 03        BCS   $B150      ;groesser als "Z" ?
 B14D   4C E6 B0     JMP   $B0E6      ;Buchstabe ?

 B150   C9 C1        CMP   #$C1       ;Shift + "A"
 B152   90 DD        BCC   $B131      ;kleiner als "Shift A" ?
 B154   C9 DB        CMP   #$DB       ;$DA = Shift + "Z"
 B156   B0 D9        BCS   $B131      ;groesser als "Shift Z" ?
 B158   4C E6 B0     JMP   $B0E6      ;Shift + Buchstabe ?

FETCHTAB:
 B15B   05 11 12 13 1C 1D 1E 1F       ;Tabelle fuer Sondercodes
 B163   90 91 92 93 94 9C 9D 9E
 B16B   9F 00 00 00

RVSBLANK:
 B170   A5 D4        LDA   $D4        ;"-Merker
 B172   48           PHA              ;sichern
 B173   A9 00        LDA   #$00
 B175   85 D4        STA   $D4        ;"-Merker loeschen
 B177   A9 12        LDA   #$12       ;(RVS on)
 B179   20 D2 FF     JSR   BSOUT      ;ausgeben
 B17C   A9 20        LDA   #$20       ;" "
 B17E   20 D2 FF     JSR   BSOUT      ;ausgeben
 B181   A9 9D        LDA   #$9D       ;(Cursor left)
 B183   20 D2 FF     JSR   BSOUT      ;ausgeben
 B186   A9 92        LDA   #$92       ;(RVS off)
 B188   20 D2 FF     JSR   BSOUT      ;ausgeben
 B18B   68           PLA
 B18C   85 D4        STA   $D4        ;"-Merker restaurieren
 B18E   60           RTS

BEFSCRSV:
 B18F   20 DB 83     JSR   INCBASBZ   ;Code ueberlesen
 B192   20 23 81     JSR   SOPEN      ;OPEN-Befehl
 B195   A6 B8        LDX   $B8        ;Dateinummer
 B197   20 C9 FF     JSR   CHKOUT     ;als Ausgabekanal
 B19A   A2 00        LDX   #$00       ;Zaehler
 B19C   BD 00 04     LDA   $0400,X    ;Video-RAM 1.Teil
 B19F   20 D2 FF     JSR   BSOUT      ;1 Byte ausgeben
 B1A2   BD 00 D8     LDA   $D800,X    ;Farb-RAM 1.Teil
 B1A5   20 D2 FF     JSR   BSOUT
 B1A8   BD FA 04     LDA   $04FA,X    ;Video-RAM 2.Teil
 B1AB   20 D2 FF     JSR   BSOUT
 B1AE   BD FA D8     LDA   $D8FA,X    ;Farb-RAM 2.Teil
 B1B1   20 D2 FF     JSR   BSOUT
 B1B4   BD F4 05     LDA   $05F4,X    ;Video-RAM 3.Teil
 B1B7   20 D2 FF     JSR   BSOUT
 B1BA   BD F4 D9     LDA   $D9F4,X    ;Farb-RAM 3.Teil
 B1BD   20 D2 FF     JSR   BSOUT
 B1C0   BD EE 06     LDA   $06EE,X    ;Video-RAM 4.Teil
```

```
 B1C3   20 D2 FF   JSR   BSOUT
 B1C6   BD EE DA   LDA   $DAEE,X   ;Farb-RAM 4.Teil
 B1C9   20 D2 FF   JSR   BSOUT
 B1CC   E8         INX             ;Zaehler erhoehen
 B1CD   E0 FA      CPX   #$FA      ;250 = 1000/4
 B1CF   D0 CB      BNE   $B19C     ;nicht fertig ?
 B1D1   20 CC FF   JSR   CLRCH     ;Kanal schliessen
 B1D4   A5 B8      LDA   $B8       ;Dateinummer
 B1D6   20 C3 FF   JSR   CLOSE     ;Datei schliessen
 B1D9   4C 2A 82   JMP   ENDSMB    ;Befehl fertig

BEFSCRLD:
 B1DC   20 DB 83   JSR   INCBASBZ ;Kommentar siehe SCRSV
 B1DF   20 23 81   JSR   SOPEN
 B1E2   A6 B8      LDX   $B8
 B1E4   20 C6 FF   JSR   CHKIN
 B1E7   A2 00      LDX   #$00
 B1E9   20 CF FF   JSR   BASIN     ;ein Zeichen holen
 B1EC   9D 00 04   STA   $0400,X
 B1EF   20 CF FF   JSR   BASIN
 B1F2   9D 00 D8   STA   $D800,X
 B1F5   20 CF FF   JSR   BASIN
 B1F8   9D FA 04   STA   $04FA,X
 B1FB   20 CF FF   JSR   BASIN
 B1FE   9D FA D8   STA   $D8FA,X
 B201   20 CF FF   JSR   BASIN
 B204   9D F4 05   STA   $05F4,X
 B207   20 CF FF   JSR   BASIN
 B20A   9D F4 D9   STA   $D9F4,X
 B20D   20 CF FF   JSR   BASIN
 B210   9D EE 06   STA   $06EE,X
 B213   20 CF FF   JSR   BASIN
 B216   9D EE DA   STA   $DAEE,X
 B219   E8         INX
 B21A   E0 FA      CPX   #$FA
 B21C   D0 CB      BNE   $B1E9
 B21E   20 CC FF   JSR   CLRCH
 B221   A5 B8      LDA   $B8
 B223   20 C3 FF   JSR   CLOSE
 B226   4C 2A 82   JMP   ENDSMB

BEFTEXT:
 B229   20 DB 83   JSR   INCBASBZ ;Code ueberlesen
 B22C   20 27 94   JSR   SGETADR  ;X holen
 B22F   84 09      STY   $09      ;als X-Koord. speichern
 B231   85 0A      STA   $0A
 B233   20 FC 81   JSR   SGETBYTC ;Y holen
 B236   86 A4      STX   $A4      ;als Y-Koord. speichern
 B238   20 31 80   JSR   SCHKCOM  ;Komma ?
 B23B   20 B7 86   JSR   SGETSTR  ;String holen
 B23E   85 21      STA   $21
```

```
B240   86 20      STX   $20        ;Stringadr. nach $20,$21
B242   A5 69      LDA   $69        ;Stringlaenge
B244   85 23      STA   $23        ;nach $23
B246   20 7D 93   JSR   GETPKTF    ;Punktfarbe holen
B249   20 FC 81   JSR   SGETBYTC   ;Groesse holen
B24C   8E 6E CB   STX   CHGR       ;Zeichengroesse
B24F   E0 00      CPX   #$00
B251   D0 05      BNE   $B258      ;nicht 0 ?
B253   A9 01      LDA   #$01
B255   8D 6E CB   STA   CHGR       ;sonst Zeichengr. = 1
B258   20 FC 81   JSR   SGETBYTC   ;Abstand holen
B25B   8E 48 CB   STX   KRX        ;KRX hier fuer Abstand
B25E   A0 00      LDY   #$00
B260   8C 4D CB   STY   SCHRART    ;Schriftarten-Flag
B263   84 6A      STY   $6A        ;RVS-Flag
B265   A5 A4      LDA   $A4        ;Koordinaten auf Stack
B267   48         PHA
B268   A5 09      LDA   $09
B26A   48         PHA
B26B   A5 0A      LDA   $0A
B26D   48         PHA
B26E   AD 6E CB   LDA   CHGR       ;Groesse auch auf Stack
B271   48         PHA
B272   B1 20      LDA   ($20),Y    ;1. Zeichen des Strings
B274   C9 12      CMP   #$12       ;(RVS on)
B276   D0 07      BNE   $B27F      ;nein ?
B278   A9 80      LDA   #$80
B27A   85 6A      STA   $6A        ;RVS-Flag setzen
B27C   4C 48 B3   JMP   $B348      ;Ende ueberpruefen

B27F   C9 92      CMP   #$92       ;(RVS off)
B281   D0 07      BNE   $B28A      ;nein ?
B283   A9 00      LDA   #$00
B285   85 6A      STA   $6A        ;RVS-Flag loeschen
B287   4C 48 B3   JMP   $B348      ;Ende ueberpruefen

B28A   C9 01      CMP   #$01       ;(CTRL-A)
B28C   D0 08      BNE   $B296      ;nein ?
B28E   A9 00      LDA   #$00
B290   8D 4D CB   STA   SCHRART    ;Schriftart = 0
B293   4C 48 B3   JMP   $B348      ;Ende ueberpruefen

B296   C9 02      CMP   #$02       ;(CTRL-B)
B298   D0 08      BNE   $B2A2      ;nein ?
B29A   A9 01      LDA   #$01
B29C   8D 4D CB   STA   SCHRART    ;Schriftart = 1
B29F   4C 48 B3   JMP   $B348      ;Ende uberpruefen

B2A2   85 AC      STA   $AC        ;Zeichen nach $AC
B2A4   C9 FF      CMP   #$FF       ;Code fuer PI
B2A6   D0 05      BNE   $B2AD      ;nein ?
```

```
B2A8  A9 7E     LDA  #$7E      ;BS-Code $7E
B2AA  4C BE B2  JMP  $B2BE     ;weiter

B2AD  C9 E0     CMP  #$E0
B2AF  90 06     BCC  $B2B7     ;ASC-Code kleiner $E0 ?
B2B1  38        SEC
B2B2  E9 40     SBC  #$40      ;$40 abziehen
B2B4  4C BE B2  JMP  $B2BE     ;weiter
B2B7  C9 C0     CMP  #$C0
B2B9  90 03     BCC  $B2BE     ;ASC-Code kleiner $C0 ?
B2BB  38        SEC
B2BC  E9 60     SBC  #$60      ;$60 abziehen

B2BE  C9 40     CMP  #$40
B2C0  90 15     BCC  $B2D7     ;Code kleiner $40 ?
B2C2  C9 60     CMP  #$60
B2C4  B0 05     BCS  $B2CB     ;Code groesser gleich $60 ?
B2C6  29 BF     AND  #$BF      ;Bit 6 ausblenden
B2C8  4C D7 B2  JMP  $B2D7     ;weiter

B2CB  C9 80     CMP  #$80
B2CD  B0 05     BCS  $B2D4     ;Code groesser gleich $80 ?
B2CF  29 DF     AND  #$DF      ;Bit 5 ausblenden
B2D1  4C D7 B2  JMP  $B2D7     ;weiter
B2D4  38        SEC
B2D5  E9 40     SBC  #$40      ;sonst $40 abziehen

B2D7  45 6A     EOR  $6A       ;RVS-Flag beruecksichtigen
B2D9  85 AC     STA  $AC       ;zu malender BS-Code
B2DB  AD 4D CB  LDA  SCHRART
B2DE  85 AD     STA  $AD       ;'Bit 8' des BS-Codes
B2E0  98        TYA            ;Y retten
B2E1  48        PHA
B2E2  20 B4 A1  JSR  CHARZEI ;Zeichen malen
B2E5  68        PLA
B2E6  A8        TAY            ;Y restaurieren
B2E7  68        PLA
B2E8  8D 6E CB  STA  CHGR      ;Groesse und Koord. rest.
B2EB  68        PLA
B2EC  85 0A     STA  $0A
B2EE  68        PLA
B2EF  85 09     STA  $09
B2F1  68        PLA
B2F2  85 A4     STA  $A4
B2F4  18        CLC
B2F5  A5 09     LDA  $09      ;$09,$0A = $09,$0A + Abstand
B2F7  6D 48 CB  ADC  KRX      ;Abstand der Zeichen
B2FA  85 09     STA  $09
B2FC  A5 0A     LDA  $0A
B2FE  69 00     ADC  #$00
B300  85 0A     STA  $0A
```

```
 B302  C8         INY             ;Zaehler im String erh.
 B303  C4 23      CPY  $23        ;mit Stringlaenge vgl.
 B305  D0 03      BNE  $B30A      ;nicht erreicht ?
 B307  4C 2A 82   JMP  ENDSMB     ;Befehl fertig

 B30A  4C 65 B2   JMP  $B265      ;zur Schleife

BEFCSET:
 B30D  20 02 82   JSR  SGETBYTN   ;Typ (0,1,2) holen
 B310  E0 02      CPX  #$02
 B312  D0 1C      BNE  $B330      ;nicht 2 ?
 B314  A9 95      LDA  #$95       ;(unsinn (s.u.))
 B316  8D 00 DD   STA  CIA2PRA
 B319  A9 3B      LDA  #$3B       ;Hires ein
 B31B  8D 11 D0   STA  VICST1     ;VIC Steuerreg. 1
 B31E  A9 0B      LDA  #$0B       ;Grafik nach $E000
 B320  8D 18 D0   STA  VICADS     ;VIC Adressreg.
 B323  A9 94      LDA  #$94       ;VIC-Bereich ab $C000
 B325  8D 00 DD   STA  CIA2PRA    ;CIA2 Port A
 B328  A9 0A      LDA  #$0A
 B32A  8D B3 C5   STA  GMEMFLAG   ;Grafik-RAM-Flag
 B32D  4C 2A 82   JMP  ENDSMB     ;Befehl fertig

 B330  20 37 A8   JSR  NRM        ;CIA2 und VIC normieren
 B333  E0 01      CPX  #$01       ;Typ von CSET in X
 B335  D0 05      BNE  $B33C      ;nicht 1 ?
 B337  A9 0E      LDA  #$0E       ;(Kleinschrift)
 B339  20 D2 FF   JSR  BSOUT      ;ausgeben
 B33C  4C 2A 82   JMP  ENDSMB     ;Befehl fertig

BEFVOL:
 B33F  20 02 82   JSR  SGETBYTN   ;Lautstaerke holen
 B342  8E 18 D4   STX  $D418      ;in SID-Register 24
 B345  4C 2A 82   JMP  ENDSMB     ;Befehl fertig

;Fortsetzung zu TEXT
 B348  68         PLA             ;Groesse und Koord. von
 B349  8D 6E CB   STA  CHGR       ;Stack holen und als akt.
 B34C  68         PLA             ;Parameter speichern
 B34D  85 0A      STA  $0A
 B34F  68         PLA
 B350  85 09      STA  $09
 B352  68         PLA
 B353  85 A4      STA  $A4
 B355  C8         INY             ;Zeahler im String erhoeh.
 B356  C4 23      CPY  $23        ;Stringlaenge
 B358  D0 B0      BNE  $B30A      ;nicht err., dann Schleife
 B35A  4C 2A 82   JMP  ENDSMB     ;Befehl fertig
```

```
BEFDISK:
 B35D   20 DB 83   JSR   INCBASBZ ;Code ueberlesen
 B360   20 CC FF   JSR   CLRCH    ;Kanaele schliessen
 B363   A9 01      LDA   #$01
 B365   20 C3 FF   JSR   CLOSE    ;Datei #1 schliessen
 B368   20 CC FF   JSR   CLRCH    ;Kanaele schliessen
 B36B   20 B7 86   JSR   SGETSTR  ;String holen
 B36E   A8         TAY            ;Stringadr. high
 B36F   A5 69      LDA   $69      ;Stringlaenge
 B371   20 BD FF   JSR   SETFNPAR ;Filenamenparameter setzen
 B374   A9 01      LDA   #$01     ;Dateinr. 1
 B376   A2 08      LDX   #$08     ;Geraetenr. 8
 B378   A0 0F      LDY   #$0F     ;Sekundaeradresse 15
 B37A   20 BA FF   JSR   SETFPAR  ;File-parameter setzen
 B37D   20 C0 FF   JSR   OPEN     ;Datei oeffnen
 B380   A2 01      LDX   #$01
 B382   20 C9 FF   JSR   CHKOUT   ;Datei #1 zur Ausgabe
 B385   20 CC FF   JSR   CLRCH    ;Kanaele schliessen
 B388   A9 01      LDA   #$01
 B38A   20 C3 FF   JSR   CLOSE    ;Datei #1 schliessen
 B38D   20 CC FF   JSR   CLRCH    ;Kanaele schliessen
 B390   4C 2A 82   JMP   ENDSMB   ;Befehl fertig

BEFUSE:
 B393   20 DB 83   JSR   INCBASBZ ;Code ueberlesen
 B396   20 B7 86   JSR   SGETSTR  ;String holen
 B399   85 A7      STA   $A7      ;Stringadr. nach $A6,$A7
 B39B   86 A6      STX   $A6
 B39D   A5 69      LDA   $69
 B39F   85 AE      STA   $AE      ;Stringlaenge nach $AE
 B3A1   A0 00      LDY   #$00
 B3A3   B1 A6      LDA   ($A6),Y  ;Zeichen aus String
 B3A5   C9 23      CMP   #$23     ;"#"
 B3A7   D0 02      BNE   $B3AB    ;nein ?
 B3A9   A9 20      LDA   #$20     ;" "
 B3AB   99 12 02   STA   USEPUFF,Y ; Zeichen in Puffer legen
 B3AE   C8         INY            ;Zaehler erhoehen
 B3AF   C4 69      CPY   $69      ;Stringlaenge
 B3B1   D0 F0      BNE   $B3A3    ;nein, dann Schleife
 B3B3   8C 0D 02   STY   USELEN   ;Strinlaenge speichern
 B3B6   20 31 80   JSR   SCHKCOM  ;Komma ?
 B3B9   20 B7 86   JSR   SGETSTR  ;String holen
 B3BC   85 A9      STA   $A9      ;Stringadr. nach $A8,$A9
 B3BE   86 A8      STX   $A8
 B3C0   A5 69      LDA   $69      ;Stringlaenge
 B3C2   85 AF      STA   $AF      ;nach $AF
 B3C4   A0 00      LDY   #$00
 B3C6   84 A4      STY   $A4      ;Position von "."in Str. 2
 B3C8   84 09      STY   $09      ;Position von "."in Str. 1
 B3CA   A2 00      LDX   #$00
 B3CC   B1 A6      LDA   ($A6),Y  ;Zeichen aus 1. String
```

```
B3CE  C9 2E      CMP  #$2E       ;"."
B3D0  F0 0C      BEQ  $B3DE      ;ja ?
B3D2  E6 09      INC  $09        ;Position erhoehen
B3D4  E8         INX
B3D5  20 E9 9A   JSR  INCA6A7    ;naechstes Zeichen
B3D8  A5 09      LDA  $09        ;Position
B3DA  C5 AE      CMP  $AE        ;1. Stringlaenge
B3DC  D0 EE      BNE  $B3CC      ;nein, dann Schleife

B3DE  A5 09      LDA  $09
B3E0  85 0A      STA  $0A        ;Punktposition 1
B3E2  8E 79 CB   STX  ZEILENANZ  ;hier 1. Punktpos.

B3E5  B1 A8      LDA  ($A8),Y    ;Zeichen aus 2.String
B3E7  C9 2E      CMP  #$2E       ;"."
B3E9  F0 0B      BEQ  $B3F6      ;ja ?
B3EB  E6 A4      INC  $A4        ;Position erhoehen
B3ED  20 F0 9A   JSR  INCA8A9    ;naechstes Zeichen
B3F0  A5 A4      LDA  $A4        ;Position
B3F2  C5 AF      CMP  $AF        ;2. Stringlaenge
B3F4  D0 EF      BNE  $B3E5      ;nicht err., dann Schleife

B3F6  A5 A4      LDA  $A4
B3F8  85 A5      STA  $A5        ;Punktposition 2
B3FA  A5 A9      LDA  $A9        ;Stringadr. 2 nach AY,AY+1
B3FC  8D AD C5   STA  AY+1
B3FF  A5 A8      LDA  $A8
B401  8D AC C5   STA  AY
B404  A5 A7      LDA  $A7        ;Stringadr. 1 nach AX,AX+1
B406  8D 98 C5   STA  AX+1
B409  A5 A6      LDA  $A6
B40B  8D 97 C5   STA  AX
B40E  20 F0 9A   JSR  INCA8A9    ;naechstes Zeichen (2)
B411  E6 A4      INC  $A4        ;Pos.Zaehler 2 erhoehen
B413  A5 09      LDA  $09        ;Positionszaehler 1
B415  C5 AE      CMP  $AE        ;1. Stringlaenge
B417  B0 1E      BCS  $B437      ;fertig mit Vorkommast. ?
B419  A5 A4      LDA  $A4        ;Positionszaehler 2
B41B  C5 AF      CMP  $AF        ;2. Stringlaenge
B41D  B0 18      BCS  $B437      ;fertig ?
B41F  B1 A6      LDA  ($A6),Y    ;Zeichen aus 1. String
B421  C9 23      CMP  #$23       ;"#"
B423  D0 07      BNE  $B42C      ;nein ?
B425  B1 A8      LDA  ($A8),Y    ;Zeichen aus 2.String
B427  20 F0 9A   JSR  INCA8A9    ;naechstes Zei. in 2. Str.
B42A  E6 A4      INC  $A4        ;Positionszaehler 2
B42C  20 9B B4   JSR  MERKZEI    ;Zeichen ablegen
B42F  E6 09      INC  $09        ;Positionszaehler 1
B431  20 E9 9A   JSR  INCA6A7    ;naechstes Zei. in 1. Str.
B434  4C 13 B4   JMP  $B413      ;Schleife
```

```
B437  AD 97 C5   LDA  AX         ;$A6,$A7 und $A8,$A9
B43A  85 A6      STA  $A6        ;restaurieren
B43C  AD 98 C5   LDA  AX+1
B43F  85 A7      STA  $A7
B441  AD AC C5   LDA  AY
B444  85 A8      STA  $A8
B446  AD AD C5   LDA  AY+1
B449  85 A9      STA  $A9
B44B  AE 79 CB   LDX  ZEILENANZ ;1. Punktpos.
B44E  A5 A8      LDA  $A8        ;$A8,$A9 um 1 vermindern
B450  D0 02      BNE  $B454
B452  C6 A9      DEC  $A9
B454  C6 A8      DEC  $A8
B456  C6 A5      DEC  $A5
B458  A5 0A      LDA  $0A        ;Position 1
B45A  C9 FF      CMP  #$FF
B45C  F0 2C      BEQ  $B48A      ;abgelaufen ?
B45E  A5 A5      LDA  $A5        ;Position 2
B460  C9 FF      CMP  #$FF
B462  F0 26      BEQ  $B48A      ;abgelaufen ?
B464  B1 A6      LDA  ($A6),Y    ;Zeichen aus 1. String
B466  C9 23      CMP  #$23       ;"#"
B468  D0 0E      BNE  $B478      ;nein ?
B46A  B1 A8      LDA  ($A8),Y    ;Zeichen aus 2. String
B46C  48         PHA             ;sichern
B46D  C6 A5      DEC  $A5        ;Position 2 vermindern
B46F  A5 A8      LDA  $A8        ;$A8,$A9 vermindern
B471  D0 02      BNE  $B475
B473  C6 A9      DEC  $A9
B475  C6 A8      DEC  $A8
B477  68         PLA             ;Zeichen wieder holen
B478  20 9B B4   JSR  MERKZEI    ;Zeichen ablegen
B47B  CA         DEX             ;X vermindern, da jetzt
B47C  CA         DEX             ;Schleife ruckwaerts
B47D  C6 0A      DEC  $0A        ;Position 1 vermindern
B47F  A5 A6      LDA  $A6        ;$A6,$A7 vermindern
B481  D0 02      BNE  $B485
B483  C6 A7      DEC  $A7
B485  C6 A6      DEC  $A6
B487  4C 58 B4   JMP  $B458      ;Schleife

;Puffer ausgeben und fertig
B48A  A2 00      LDX  #$00
B48C  BD 12 02   LDA  USEPUFF,X ;Zeichen aus Puffer
B48F  20 D2 FF   JSR  BSOUT      ;ausgeben
B492  E8         INX
B493  EC 0D 02   CPX  USELEN     ;auszugebende Laenge
B496  D0 F4      BNE  $B48C      ;nicht err., dann Schleife
B498  4C 2A 82   JMP  ENDSMB     ;Befehl fertig
```

```
MERKZEI:
 B49B   9D 12 02    STA   USEPUFF,X ;Zeichen in Puffer legen
 B49E   E8          INX             ;Zeiger erhoehen
 B49F   60          RTS

BEFHRDCPY:
 B4A0   20 DB 83    JSR   INCBASBZ  ;Code ueberlesen
 B4A3   20 CC FF    JSR   CLRCH     ;Kanaele schliessen
 B4A6   A9 01       LDA   #$01      ;Datei #1
 B4A8   20 C3 FF    JSR   CLOSE     ;schliessen
 B4AB   20 CC FF    JSR   CLRCH     ;Kanaele schliessen
 B4AE   A9 01       LDA   #$01      ;Datei #1
 B4B0   A2 04       LDX   #$04      ;Geraetenr. 4
 B4B2   A0 FF       LDY   #$FF      ;keine Sek. adr.
 B4B4   20 BA FF    JSR   SETFPAR   ;Fileparameter setzen
 B4B7   20 C0 FF    JSR   OPEN      ;Datei oeffnen
 B4BA   A2 01       LDX   #$01
 B4BC   20 C9 FF    JSR   CHKOUT    ;Datei #1 = Ausgabekanal
 B4BF   A9 0D       LDA   #$0D      ;Zeilenvorschub
 B4C1   20 D2 FF    JSR   BSOUT     ;ausgeben
 B4C4   A9 04       LDA   #$04      ;$20,$21 auf $0400
 B4C6   85 21       STA   $21
 B4C8   A9 00       LDA   #$00
 B4CA   85 20       STA   $20
 B4CC   A0 00       LDY   #$00
 B4CE   A2 00       LDX   #$00
 B4D0   B1 20       LDA   ($20),Y   ;Zeichen aus Video-RAM
 B4D2   C9 80       CMP   #$80
 B4D4   90 09       BCC   $B4DF     ;BS-Code kleiner 128 ?
 B4D6   49 80       EOR   #$80      ;Bit 7 loeschen
 B4D8   48          PHA             ;Wert sichern
 B4D9   A9 12       LDA   #$12      ;(RVS on)
 B4DB   20 D2 FF    JSR   BSOUT     ;ausgeben
 B4DE   68          PLA             ;Wert holen
 B4DF   C9 20       CMP   #$20      ;" "
 B4E1   B0 13       BCS   $B4F6     ;groesser gleich " " ?
 B4E3   48          PHA             ;Wert wieder auf Stapel
 B4E4   AD 18 D0    LDA   VICADS    ;VIC Adressregister
 B4E7   C9 17       CMP   #$17      ;Wert bei Gr./Kleinschr.
 B4E9   D0 05       BNE   $B4F0     ;Gross/Grafikz. ?
 B4EB   A9 11       LDA   #$11      ;(Cursor down) (Keins.)
 B4ED   20 D2 FF    JSR   BSOUT     ;ausgeben
 B4F0   68          PLA             ;Code holen
 B4F1   09 40       ORA   #$40      ;Bit 6
 B4F3   4C 18 B5    JMP   $B518     ;weiter

 B4F6   C9 40       CMP   #$40
 B4F8   90 1D       BCC   $B517     ;kleiner 64 ?
 B4FA   C9 7F       CMP   #$7F
 B4FC   F0 14       BEQ   $B512     ;Code = 127 ?
```

```
B4FE  48          PHA            ;Code sichern
B4FF  AD 18 D0    LDA  VICADS    ;VIC-Adressrgister
B502  C9 17       CMP  #$17      ;Gross-Kleinschrift
B504  D0 05       BNE  $B50B     ;nein ?
B506  A9 11       LDA  #$11      ;(Umsch. auf Kleinschr.)
B508  20 D2 FF    JSR  BSOUT     ;ausgeben
B50B  68          PLA            ;Code holen
B50C  18          CLC
B50D  69 80       ADC  #$80      ;+ 128
B50F  4C 18 B5    JMP  $B518     ;weiter

B512  A9 BF       LDA  #$BF      ;Code = 191
B514  4C 18 B5    JMP  $B518     ;weiter

B517  EA          NOP            ;(???)
B518  C9 22       CMP  #$22      ;"
B51A  D0 2B       BNE  $B547     ;kein Anf.zeichen ?
B51C  A9 08       LDA  #$08      ;Codes 8,128,131,128,128,
B51E  20 D2 FF    JSR  BSOUT     ;131,128,15 ausgeben
B521  A9 80       LDA  #$80      ;Bit-Image-Sequenz fuer
B523  20 D2 FF    JSR  BSOUT     ;die Darstellung eines
B526  A9 83       LDA  #$83      ;Anfuehrungszeichens (")
B528  20 D2 FF    JSR  BSOUT
B52B  A9 80       LDA  #$80
B52D  20 D2 FF    JSR  BSOUT
B530  A9 80       LDA  #$80
B532  20 D2 FF    JSR  BSOUT
B535  A9 83       LDA  #$83
B537  20 D2 FF    JSR  BSOUT
B53A  A9 80       LDA  #$80
B53C  20 D2 FF    JSR  BSOUT
B53F  A9 0F       LDA  #$0F
B541  20 D2 FF    JSR  BSOUT
B544  4C 4A B5    JMP  $B54A     ;weiter

B547  20 D2 FF    JSR  BSOUT     ;Code ausgeben
B54A  A9 00       LDA  #$00
B54C  85 D4       STA  $D4       ;"-Merker = 0
B54E  A9 91       LDA  #$91      ;(Cursor up) = (Grafikz.)
B550  20 D2 FF    JSR  BSOUT     ;ausgeben
B553  A9 92       LDA  #$92      ;(RVS off)
B555  20 D2 FF    JSR  BSOUT     ;ausgeben
B558  C8          INY            ;naechstes Zeichen
B559  C0 28       CPY  #$28
B55B  F0 03       BEQ  $B560     ;Zeilenende ?
B55D  4C D0 B4    JMP  $B4D0     ;Schleife

B560  A9 0D       LDA  #$0D      ;Zeilenvorschub
B562  20 D2 FF    JSR  BSOUT     ;ausgeben
B565  A0 00       LDY  #$00
B567  18          CLC
```

```
 B568   A5 20       LDA  $20       ;$20,$21 um 40 erhoehen
 B56A   69 28       ADC  #$28
 B56C   85 20       STA  $20
 B56E   A5 21       LDA  $21
 B570   69 00       ADC  #$00
 B572   85 21       STA  $21
 B574   E8          INX            ;naechste Zeile
 B575   E0 19       CPX  #$19      ;25 Zeilen
 B577   D0 E2       BNE  $B55B     ;nicht fertig ?
 B579   20 CC FF    JSR  CLRCH     ;Kanaele schliessen
 B57C   A9 01       LDA  #$01
 B57E   20 C3 FF    JSR  CLOSE     ;Datei #1 schliessen
 B581   20 CC FF    JSR  CLRCH     ;Kanaele schliessen
 B584   4C 2A 82    JMP  ENDSMB    ;Befehl fertig

;bei 'KEY128'
 B587   20 31 80    JSR  SCHKCOM   ;Komma ?
 B58A   4C AA BD    JMP  DSCBMOUT1 ;'ds-cbm' ausgeben (???)

;bei 'KEY0'
 B58D   4C 16 BE    JMP  $BE16     ;zum DISPLAY-Befehl

BEFKEY:
 B590   20 02 82    JSR  SGETBYTN  ;Nr. holen
 B593   E0 00       CPX  #$00
 B595   F0 F6       BEQ  $B58D     ;0 ?
 B597   E0 80       CPX  #$80
 B599   F0 EC       BEQ  $B587     ;128 ?
 B59B   E0 11       CPX  #$11
 B59D   90 05       BCC  $B5A4     ;kleiner 17 ?
 B59F   A9 0C       LDA  #$0C      ;nr. f. 'bad mode'
 B5A1   4C 8C 88    JMP  SERROUT   ;Fehler ausgeben

 B5A4   CA          DEX
 B5A5   86 AA       STX  $AA       ;Nr.-1 nach $AA
 B5A7   A9 00       LDA  #$00
 B5A9   85 AB       STA  $AB       ;$AA,$AB = 16 * (Nr.-1)
 B5AB   06 AA       ASL  $AA
 B5AD   26 AB       ROL  $AB
 B5AF   06 AA       ASL  $AA
 B5B1   26 AB       ROL  $AB
 B5B3   06 AA       ASL  $AA
 B5B5   26 AB       ROL  $AB
 B5B7   06 AA       ASL  $AA
 B5B9   26 AB       ROL  $AB
 B5BB   18          CLC
 B5BC   A5 AA       LDA  $AA       ;$AA,$AB = $AA,$AB + $C64D
 B5BE   69 4D       ADC  #$4D
 B5C0   85 AA       STA  $AA
 B5C2   A5 AB       LDA  $AB
 B5C4   69 C6       ADC  #$C6
```

```
 B5C6   85 AB       STA  $AB
 B5C8   20 31 80    JSR  SCHKCOM   ;Komma ?
 B5CB   20 B7 86    JSR  SGETSTR   ;String holen
 B5CE   85 A9       STA  $A9       ;Stringadr. nach $A8,$A9
 B5D0   86 A8       STX  $A8
 B5D2   A0 00       LDY  #$00
 B5D4   B1 A8       LDA  ($A8),Y   ;Zeichen aus String
 B5D6   91 AA       STA  ($AA),Y   ;in Key-Tabelle
 B5D8   C8          INY
 B5D9   C4 69       CPY  $69       ;Stringlaenge
 B5DB   F0 04       BEQ  $B5E1     ;erreicht ?
 B5DD   C0 0F       CPY  #$0F      ;max. 15 Zeichen
 B5DF   90 F3       BCC  $B5D4     ;nicht erreicht ?
 B5E1   A9 00       LDA  #$00      ;Tabelleneintrag mit 0 ab-
 B5E3   91 AA       STA  ($AA),Y   ;schliessen
 B5E5   4C 2A 82    JMP  ENDSMB    ;Befehl fertig

BEFPAINT:
 B5E8   20 DB 83    JSR  INCBASBZ ;Code ueberlesen
 B5EB   20 27 94    JSR  SGETADR   ;Start X holen
 B5EE   8D 98 C5    STA  AX+1      ;nach AX,AX+1 und $09,$0A
 B5F1   85 0A       STA  $0A
 B5F3   8C 97 C5    STY  AX
 B5F6   84 09       STY  $09
 B5F8   20 FC 81    JSR  SGETBYTC ;Start Y holen
 B5FB   8E AC C5    STX  AY        ;nach AY und $A4
 B5FE   86 A4       STX  $A4
 B600   20 7D 93    JSR  GETPKTF   ;Punktfarbe holen
 B603   A9 00       LDA  #$00
 B605   85 65       STA  $65       ;div. Flags annullieren
 B607   85 64       STA  $64
 B609   85 62       STA  $62
 B60B   85 59       STA  $59
 B60D   85 57       STA  $57
 B60F   85 58       STA  $58
 B611   85 5A       STA  $5A
 B613   A5 F7       LDA  $F7         ;Punktfarbe
 B615   85 66       STA  $66         ;in $66 sichern
 B617   A9 0B       LDA  #$0B        ;Punktfarbe 11 (TEST)
 B619   85 F7       STA  $F7
 B61B   20 97 92    JSR  PUNKT       ;Punkt gesetzt ?
 B61E   A5 90       LDA  $90         ;Ergebnis von 'TEST'
 B620   85 6A       STA  $6A         ;Ist-Punktfarbe
 B622   C5 66       CMP  $66         ;Soll-Punktfarbe
 B624   D0 03       BNE  $B629       ;Punkt nicht gesetzt ?
 B626   4C 2A 82    JMP  ENDSMB      ;sonst Befehl fertig

 B629   AE 97 C5    LDX  AX      ;$5F,$60 und $61 als akt.
 B62C   86 5F       STX  $5F     ;Koordinaten; im Folgenden
 B62E   AD 98 C5    LDA  AX+1    ;nur als x und y bezeichnet
 B631   85 60       STA  $60
```

```
B633  AC AC C5   LDY  AY
B636  84 61      STY  $61
B638  38         SEC
B639  A5 5F      LDA  $5F       ;$AA,$AB = x - 1
B63B  E9 01      SBC  #$01
B63D  85 AA      STA  $AA
B63F  A5 60      LDA  $60
B641  E9 00      SBC  #$00
B643  85 AB      STA  $AB
B645  A5 AB      LDA  $AB
B647  A6 AA      LDX  $AA
B649  A4 61      LDY  $61
B64B  20 BA B9   JSR  TESTFREI  ;test(x-1,y)
B64E  D0 0B      BNE  $B65B     ;Punkt gesetzt ?
B650  A5 5F      LDA  $5F       ;x = x - 1
B652  D0 02      BNE  $B656
B654  C6 60      DEC  $60
B656  C6 5F      DEC  $5F
B658  4C 38 B6   JMP  $B638      ;Schleife

B65B  A5 5F      LDA  $5F       ;naechst linken Randpunkt
B65D  8D 97 C5   STA  AX        ;als Anfangspunkt speichern
B660  A5 60      LDA  $60
B662  8D 98 C5   STA  AX+1
B665  4C 24 B9   JMP  $B924     ;s.u.

B668  A5 5B      LDA  $5B       ;Flagsatz $5B,$5C,$5D,$5E
B66A  85 57      STA  $57       ;nach $57,$58,$59,$5A
B66C  A5 5C      LDA  $5C       ;Flagsatz beschr. Umgebung
B66E  85 58      STA  $58       ;$5B ($57) fuer (x,y-1)
B670  A5 5D      LDA  $5D       ;$5C ($58) fuer (x,y+1)
B672  85 59      STA  $59       ;$5D ($59) fuer (x+1,y)
B674  A5 5E      LDA  $5E       ;$5E ($5A) fuer (x-1,y)
B676  85 5A      STA  $5A
B678  A9 01      LDA  #$01
B67A  85 5B      STA  $5B       ;$5B und $5C auf 1 setzen
B67C  85 5C      STA  $5C
B67E  AD 2C C5   LDA  HILFFLAG
B681  C9 0A      CMP  #$0A
B683  F0 13      BEQ  $B698     ;Hilfflag = $0A ?
B685  A6 5F      LDX  $5F
B687  A5 60      LDA  $60
B689  A4 61      LDY  $61
B68B  88         DEY
B68C  20 BA B9   JSR  TESTFREI
B68F  85 5B      STA  $5B        ;$5B = test (x,y-1)
B691  AD 2C C5   LDA  HILFFLAG
B694  C9 05      CMP  #$05
B696  F0 0C      BEQ  $B6A4      ;Hilfflag = $05 ?
B698  A6 5F      LDX  $5F
B69A  A5 60      LDA  $60
```

```
B69C  A4 61      LDY  $61
B69E  C8         INY
B69F  20 BA B9   JSR  TESTFREI
B6A2  85 5C      STA  $5C       ;$5C = test (x,y+1)
B6A4  E6 5F      INC  $5F
B6A6  D0 02      BNE  $B6AA
B6A8  E6 60      INC  $60
B6AA  A6 5F      LDX  $5F
B6AC  A5 60      LDA  $60
B6AE  A4 61      LDY  $61
B6B0  20 BA B9   JSR  TESTFREI
B6B3  85 5D      STA  $5D       ;$5D = test (x+1,y)
B6B5  38         SEC
B6B6  A5 5F      LDA  $5F
B6B8  E9 02      SBC  #$02
B6BA  85 5F      STA  $5F
B6BC  A5 60      LDA  $60
B6BE  E9 00      SBC  #$00
B6C0  85 60      STA  $60
B6C2  A6 5F      LDX  $5F
B6C4  A4 61      LDY  $61
B6C6  20 BA B9   JSR  TESTFREI
B6C9  85 5E      STA  $5E       ;$5E = test (x-1,y)
B6CB  E6 5F      INC  $5F       ;x restaurieren
B6CD  D0 02      BNE  $B6D1
B6CF  E6 60      INC  $60
B6D1  18         CLC
B6D2  A5 5B      LDA  $5B       ;Flagsumme $5B,$5C,$5D,$5E
B6D4  65 5C      ADC  $5C       ;bilden
B6D6  65 5D      ADC  $5D
B6D8  65 5E      ADC  $5E
B6DA  C9 04      CMP  #$04
B6DC  D0 03      BNE  $B6E1     ;nicht alle Flags ges. ?
B6DE  4C 3C B7   JMP  $B73C     ;sonst nach $B73C

B6E1  A5 5B      LDA  $5B       ;neues Flag (x,y-1)
B6E3  C5 57      CMP  $57       ;(invertiertes) altes
B6E5  D0 46      BNE  $B72D     ;ungleich ?
B6E7  A5 5C      LDA  $5C       ;(x,y+1)
B6E9  C5 58      CMP  $58
B6EB  D0 32      BNE  $B71F     ;ungleich
B6ED  A5 5D      LDA  $5D       ;(x+1,y)
B6EF  C5 59      CMP  $59
B6F1  D0 29      BNE  $B71C     ;ungleich ?
B6F3  A5 5E      LDA  $5E       ;(x-1,y)
B6F5  C5 5A      CMP  $5A
B6F7  D0 23      BNE  $B71C     ;ungleich ?
B6F9  A9 01      LDA  #$01
B6FB  85 62      STA  $62       ;fertig-Flag setzen
```

```
B6FD   A5 5B       LDA  $5B
B6FF   D0 05       BNE  $B706
B701   C6 61       DEC  $61
B703   4C B1 B8    JMP  $B8B1

B706   A5 5C       LDA  $5C
B708   F0 09       BEQ  $B713
B70A   A5 5B       LDA  $5B
B70C   49 01       EOR  #$01
B70E   85 57       STA  $57
B710   4C 2D B7    JMP  $B72D

B713   A9 00       LDA  #$00
B715   85 62       STA  $62        ;fertig-Flag
B717   E6 61       INC  $61
B719   4C B1 B8    JMP  $B8B1

B71C   4C 0C B8    JMP  $B80C

B71F   A5 5C       LDA  $5C
B721   D0 04       BNE  $B727
B723   A5 58       LDA  $58
B725   D0 DF       BNE  $B706
B727   A5 5B       LDA  $5B
B729   D0 11       BNE  $B73C
B72B   F0 CC       BEQ  $B6F9

B72D   A5 5B       LDA  $5B
B72F   D0 07       BNE  $B738
B731   A5 57       LDA  $57
B733   F0 03       BEQ  $B738
B735   4C DD B8    JMP  $B8DD

B738   A5 5C       LDA  $5C
B73A   F0 CA       BEQ  $B706
B73C   A5 5D       LDA  $5D
B73E   D0 2F       BNE  $B76F

B740   38          SEC
B741   A5 5F       LDA  $5F
B743   E9 01       SBC  #$01
B745   AA          TAX
B746   A5 60       LDA  $60
B748   E9 00       SBC  #$00
B74A   A4 61       LDY  $61
B74C   20 D7 B9    JSR  MERKEPKT   ;merke (x-1,y)

B74F   E6 5F       INC  $5F
B751   D0 02       BNE  $B755
B753   E6 60       INC  $60
B755   18          CLC
```

```
B756   A5 5F       LDA   $5F
B758   69 01       ADC   #$01
B75A   AA          TAX
B75B   A5 60       LDA   $60
B75D   69 00       ADC   #$00
B75F   A4 61       LDY   $61
B761   20 D7 B9    JSR   MERKEPKT ;merke (x+1,y)

B764   A5 61       LDA   $61
B766   8D 4F CA    STA   PYMERK
B769   20 6E B9    JSR   TABSAUBER
B76C   4C 0C B8    JMP   $B80C

B76F   A5 5E       LDA   $5E
B771   D0 30       BNE   $B7A3

B773   18          CLC
B774   A5 5F       LDA   $5F
B776   69 01       ADC   #$01
B778   AA          TAX
B779   A5 60       LDA   $60
B77B   69 00       ADC   #$00
B77D   A4 61       LDY   $61
B77F   20 D7 B9    JSR   MERKEPKT ;merke (x+1,y)

B782   A5 5F       LDA   $5F
B784   D0 02       BNE   $B788
B786   C6 60       DEC   $60
B788   C6 5F       DEC   $5F
B78A   38          SEC
B78B   A5 5F       LDA   $5F
B78D   E9 01       SBC   #$01
B78F   85 AA       STA   $AA
B791   A5 60       LDA   $60
B793   E9 00       SBC   #$00
B795   85 AB       STA   $AB
B797   A5 AB       LDA   $AB
B799   A6 AA       LDX   $AA
B79B   A4 61       LDY   $61
B79D   20 D7 B9    JSR   MERKEPKT ;merke (x-1,y)
B7A0   4C 64 B7    JMP   $B764

B7A3   AD 4F CA    LDA   PYMERK
B7A6   C5 61       CMP   $61
B7A8   F0 1D       BEQ   $B7C7
B7AA   A5 61       LDA   $61
B7AC   CD 4F CA    CMP   PYMERK
B7AF   F0 09       BEQ   $B7BA
B7B1   B0 0A       BCS   $B7BD
B7B3   A9 0A       LDA   #$0A
B7B5   8D 2B C5    STA   HILFFL1
```

```
 B7B8   E6 61      INC  $61
 B7BA   4C 24 B9   JMP  $B924

 B7BD   A9 05      LDA  #$05
 B7BF   8D 2B C5   STA  HILFFL1
 B7C2   C6 61      DEC  $61
 B7C4   4C 24 B9   JMP  $B924

;Punkt vom Punkt-Stapel holen und bearbeiten
 B7C7   A6 65      LDX  $65       ;Stapelzeiger
 B7C9   F0 20      BEQ  $B7EB     ;Stapel leer ?
 B7CB   CA         DEX
 B7CC   86 65      STX  $65       ;Stapelz. = Stapelz. - 1
 B7CE   BD 51 C7   LDA  PXHTAB,X ;x,y = Werte aus Stapel
 B7D1   85 60      STA  $60
 B7D3   BD 50 C8   LDA  PXLTAB,X
 B7D6   85 5F      STA  $5F
 B7D8   BD 4F C9   LDA  PYTAB,X
 B7DB   85 61      STA  $61
 B7DD   A6 5F      LDX  $5F
 B7DF   A5 60      LDA  $60
 B7E1   A4 61      LDY  $61
 B7E3   20 BA B9   JSR  TESTFREI ;test(x,y)
 B7E6   D0 DF      BNE  $B7C7     ;Pkt. ges., dann naechsten
 B7E8   4C 24 B9   JMP  $B924     ;Punkt weiter bearbeiten

 B7EB   AD AC C5   LDA  AY        ;'Anfangswert Y'
 B7EE   C9 FF      CMP  #$FF      ;kleiner 0
 B7F0   D0 03      BNE  $B7F5     ;nein ?
 B7F2   4C 2A 82   JMP  ENDSMB    ;Befehl fertig

 B7F5   AD 97 C5   LDA  AX        ;x = AX,AX+1
 B7F8   85 5F      STA  $5F
 B7FA   AD 98 C5   LDA  AX+1
 B7FD   85 60      STA  $60
 B7FF   AD AC C5   LDA  AY        ;y = AY
 B802   85 61      STA  $61
 B804   A9 FF      LDA  #$FF      ;Merker f. Y fertig
 B806   8D AC C5   STA  AY        ;setzen
 B809   4C 24 B9   JMP  $B924     ;Punkt weiter behandeln

 B80C   A5 5D      LDA  $5D
 B80E   F0 1B      BEQ  $B82B
 B810   A5 59      LDA  $59
 B812   D0 2A      BNE  $B83E
 B814   A5 62      LDA  $62       ;fertig-Flag
 B816   D0 26      BNE  $B83E

 B818   18         CLC
 B819   A5 5F      LDA  $5F
 B81B   69 01      ADC  #$01
```

```
B81D  AA          TAX
B81E  A5 60       LDA  $60
B820  69 00       ADC  #$00
B822  A4 61       LDY  $61
B824  88          DEY
B825  20 D7 B9    JSR  MERKEPKT ;merke (x+1,y-1)
B828  4C 3E B8    JMP  $B83E

B82B  A5 59       LDA  $59
B82D  F0 0F       BEQ  $B83E

B82F  18          CLC
B830  A5 5F       LDA  $5F
B832  69 01       ADC  #$01
B834  AA          TAX
B835  A5 60       LDA  $60
B837  69 00       ADC  #$00
B839  A4 61       LDY  $61
B83B  20 D7 B9    JSR  MERKEPKT ;merke (x+1,y)

B83E  A5 62       LDA  $62      ;fertig-Flag
B840  F0 1B       BEQ  $B85D
B842  A5 59       LDA  $59
B844  D0 32       BNE  $B878
B846  A5 5D       LDA  $5D
B848  F0 2E       BEQ  $B878

B84A  18          CLC
B84B  A5 5F       LDA  $5F
B84D  69 01       ADC  #$01
B84F  AA          TAX
B850  A5 60       LDA  $60
B852  69 00       ADC  #$00
B854  A4 61       LDY  $61
B856  C8          INY
B857  20 D7 B9    JSR  MERKEPKT ;merke (x+1,y+1)
B85A  4C 78 B8    JMP  $B878

B85D  A5 5E       LDA  $5E
B85F  F0 1B       BEQ  $B87C
B861  A5 5A       LDA  $5A
B863  D0 2A       BNE  $B88F

B865  38          SEC
B866  A5 5F       LDA  $5F
B868  E9 01       SBC  #$01
B86A  AA          TAX
B86B  A5 60       LDA  $60
B86D  E9 00       SBC  #$00
B86F  A4 61       LDY  $61
B871  88          DEY
```

```
B872  20 D7 B9  JSR  MERKEPKT ;merke (x-1,y-1)
B875  4C 8F B8  JMP  $B88F

B878  A5 5E     LDA  $5E
B87A  D0 13     BNE  $B88F
B87C  A5 5A     LDA  $5A
B87E  F0 0F     BEQ  $B88F

B880  38        SEC
B881  A5 5F     LDA  $5F
B883  E9 01     SBC  #$01
B885  AA        TAX
B886  A5 60     LDA  $60
B888  E9 00     SBC  #$00
B88A  A4 61     LDY  $61
B88C  20 D7 B9  JSR  MERKEPKT ;merke (x-1,y)

B88F  A5 5E     LDA  $5E
B891  C5 5A     CMP  $5A
B893  F0 1C     BEQ  $B8B1
B895  A5 5E     LDA  $5E
B897  F0 18     BEQ  $B8B1
B899  A5 5A     LDA  $5A
B89B  D0 14     BNE  $B8B1
B89D  A5 62     LDA  $62      ;fertig-Flag
B89F  F0 10     BEQ  $B8B1

B8A1  38        SEC
B8A2  A5 5F     LDA  $5F
B8A4  E9 01     SBC  #$01
B8A6  AA        TAX
B8A7  A5 60     LDA  $60
B8A9  E9 00     SBC  #$00
B8AB  A4 61     LDY  $61
B8AD  C8        INY
B8AE  20 D7 B9  JSR  MERKEPKT ;merke (x-1,y+1)

B8B1  A5 5F     LDA  $5F      ;x,y nach $09,$0A und $A4
B8B3  85 09     STA  $09
B8B5  A5 60     LDA  $60
B8B7  85 0A     STA  $0A
B8B9  A5 61     LDA  $61
B8BB  85 A4     STA  $A4
B8BD  AD 2B C5  LDA  HILFFL1
B8C0  8D 2C C5  STA  HILFFLAG
B8C3  A9 00     LDA  #$00
B8C5  8D 2B C5  STA  HILFFL1
B8C8  A5 66     LDA  $66      ;Sollfarbe
B8CA  85 F7     STA  $F7
B8CC  20 97 92  JSR  PUNKT    ;Punkt setzen
B8CF  AD 51 CA  LDA  PAKTY
```

```
B8D2  8D 50 CA   STA  PYALT
B8D5  A5 61      LDA  $61
B8D7  8D 51 CA   STA  PAKTY
B8DA  4C 68 B6   JMP  $B668

B8DD  A5 5C      LDA  $5C
B8DF  C5 58      CMP  $58
B8E1  D0 07      BNE  $B8EA
B8E3  A9 00      LDA  #$00
B8E5  85 64      STA  $64
B8E7  4C F9 B6   JMP  $B6F9

B8EA  A5 61      LDA  $61
B8EC  CD 51 CA   CMP  PAKTY
B8EF  D0 04      BNE  $B8F5
B8F1  A5 64      LDA  $64
B8F3  F0 07      BEQ  $B8FC
B8F5  A9 00      LDA  #$00
B8F7  85 64      STA  $64
B8F9  4C 1F B7   JMP  $B71F

B8FC  A5 60      LDA  $60            ;Punkt setzen bei (x,y)
B8FE  85 0A      STA  $0A
B900  A5 5F      LDA  $5F
B902  85 09      STA  $09
B904  A5 61      LDA  $61
B906  85 A4      STA  $A4
B908  A5 6A      LDA  $6A
B90A  85 F7      STA  $F7
B90C  20 97 92   JSR  PUNKT

B90F  C6 61      DEC  $61       ;y = y - 1
B911  A9 0A      LDA  #$0A
B913  85 64      STA  $64
B915  E6 61      INC  $61       ;y = y + 1
B917  A5 60      LDA  $60
B919  A6 5F      LDX  $5F
B91B  A4 61      LDY  $61
B91D  20 BA B9   JSR  TESTFREI  ;test(x,y)
B920  F0 F3      BEQ  $B915     ;frei ?, dann Schleife
B922  C6 61      DEC  $61       ;y = y - 1
B924  A5 61      LDA  $61
B926  8D 4F CA   STA  PYMERK    ;y merken

B929  A5 60      LDA  $60
B92B  A6 5F      LDX  $5F
B92D  A4 61      LDY  $61
B92F  88         DEY
B930  20 BA B9   JSR  TESTFREI  ;test(x,y-1)
B933  49 01      EOR  #$01      ;invertieren
B935  85 5B      STA  $5B       ;$5B = 1 - test(x,y-1)
```

```
 B937   A5 60      LDA  $60
 B939   A6 5F      LDX  $5F
 B93B   A4 61      LDY  $61
 B93D   C8         INY
 B93E   20 BA B9   JSR  TESTFREI
 B941   49 01      EOR  #$01
 B943   85 5C      STA  $5C       ;$5C = 1 - test(x,y+1)
 B945   18         CLC
 B946   A5 5F      LDA  $5F
 B948   69 01      ADC  #$01
 B94A   AA         TAX
 B94B   A5 60      LDA  $60
 B94D   69 00      ADC  #$00
 B94F   A4 61      LDY  $61
 B951   20 BA B9   JSR  TESTFREI
 B954   49 01      EOR  #$01
 B956   85 5D      STA  $5D       ;$5D = 1 - test(x+1,y)
 B958   38         SEC
 B959   A5 5F      LDA  $5F
 B95B   E9 01      SBC  #$01
 B95D   AA         TAX
 B95E   A5 60      LDA  $60
 B960   E9 00      SBC  #$00
 B962   A4 61      LDY  $61
 B964   20 BA B9   JSR  TESTFREI
 B967   49 01      EOR  #$01
 B969   85 5E      STA  $5E       ;$5E = 1 - test(x-1,y)
 B96B   4C B1 B8   JMP  $B8B1

TABSAUBER:
 B96E   A5 65      LDA  $65       ;Stapelzeiger
 B970   C9 14      CMP  #$14
 B972   B0 01      BCS  TABSAUBER1 ;groesser= 20 ?
 B974   60         RTS            ;sonst fertig

TABSAUBER1:
 B975   A9 00      LDA  #$00
 B977   8D 4E CA   STA  PZEIG     ;zeiger in Punktstapel
 B97A   AE 4E CA   LDX  PZEIG
 B97D   BD 4F C9   LDA  PYTAB,X
 B980   A8         TAY
 B981   BD 50 C8   LDA  PXLTAB,X
 B984   85 AA      STA  $AA
 B986   BD 51 C7   LDA  PXHTAB,X
 B989   A6 AA      LDX  $AA
 B98B   20 BA B9   JSR  TESTFREI ;test(PZEIG)
 B98E   F0 1F      BEQ  $B9AF     ;frei, dann weiter

 B990   AE 4E CA   LDX  PZEIG     ;Wert (PZEIG) aus Stapel
 B993   BD 51 C8   LDA  PXLTAB+1,X ;entfernen
 B996   9D 50 C8   STA  PXLTAB,X
```

```
 B999   BD 52 C7    LDA   PXHTAB+1,X
 B99C   9D 51 C7    STA   PXHTAB,X
 B99F   BD 50 C9    LDA   PYTAB+1,X
 B9A2   9D 4F C9    STA   PYTAB,X
 B9A5   E8          INX
 B9A6   E4 65       CPX   $65        ;Zeiger auf Tabellenende
 B9A8   D0 E9       BNE   $B993      ;nicht erreicht ?
 B9AA   C6 65       DEC   $65        ;Tabelle um 1 verkleinern
 B9AC   4C B2 B9    JMP   $B9B2      ;weiter

 B9AF   EE 4E CA    INC   PZEIG     ;naechsten Eintrag anpeilen
 B9B2   AD 4E CA    LDA   PZEIG
 B9B5   C5 65       CMP   $65
 B9B7   D0 C1       BNE   $B97A     ;Tabelle nicht fertig ?
 B9B9   60          RTS             ;Ende von TABSAUBER

TESTFREI:
 B9BA   86 09       STX   $09     ;X-Koord. in X-Reg. und Akku
 B9BC   85 0A       STA   $0A
 B9BE   84 A4       STY   $A4     ;Y-Koord. in Y-Reg.
 B9C0   A9 0B       LDA   #$0B
 B9C2   85 F7       STA   $F7     ;Punktfarbe 11 (TEST)
 B9C4   20 97 92    JSR   PUNKT   ;Punkt testen
 B9C7   A5 90       LDA   $90     ;Ergebnis
 B9C9   C5 6A       CMP   $6A     ;Farbe des letzten Nachbarn
 B9CB   F0 05       BEQ   $B9D2   ;Punkt frei ?
 B9CD   A9 01       LDA   #$01    ;Ergebnis = 1 (besetzt)
 B9CF   85 90       STA   $90
 B9D1   60          RTS

 B9D2   A9 00       LDA   #$00    ;Ergebnis = 0 (frei)
 B9D4   85 90       STA   $90
 B9D6   60          RTS

MERKEPKT:
 B9D7   86 AE       STX   $AE        ;Koordinaten merken
 B9D9   85 AF       STA   $AF
 B9DB   84 A6       STY   $A6
 B9DD   A6 65       LDX   $65        ;Zeiger auf Tabellenende
 B9DF   A5 A6       LDA   $A6
 B9E1   9D 4F C9    STA   PYTAB,X   ;Y eintragen
 B9E4   A5 AE       LDA   $AE
 B9E6   9D 50 C8    STA   PXLTAB,X  ;X eintragen
 B9E9   A5 AF       LDA   $AF
 B9EB   9D 51 C7    STA   PXHTAB,X
 B9EE   E6 65       INC   $65        ;Tabelle um 1 erweitern
 B9F0   60          RTS

BEFCOPY:
 B9F1   20 DB 83    JSR   INCBASBZ ;Code ueberlesen
 B9F4   AD 2D C5    LDA   XMAXLOW
```

```
B9F7  48        PHA               ;X-Grenzen auf Stack
B9F8  AD 2E C5  LDA  XMAXHIGH
B9FB  48        PHA
B9FC  A9 01     LDA  #$01         ;X-Grenze auf 319
B9FE  8D 2E C5  STA  XMAXHIGH
BA01  A9 3F     LDA  #$3F
BA03  8D 2D C5  STA  XMAXLOW
BA06  20 CC FF  JSR  CLRCH        ;Kanaele schliessen
BA09  A9 01     LDA  #$01
BA0B  20 C3 FF  JSR  CLOSE        ;Datei #1 schliessen
BA0E  20 CC FF  JSR  CLRCH        ;Kanaele schliessen
BA11  AD B0 C5  LDA  MULTIJN      ;Multi-Flag
BA14  48        PHA               ;auf Stack
BA15  A9 00     LDA  #$00
BA17  8D B0 C5  STA  MULTIJN      ;Multi-Modus = 0
BA1A  A2 04     LDX  #$04         ;Geraetenr. 4
BA1C  A9 01     LDA  #$01         ;Datei #1
BA1E  A0 FF     LDY  #$FF         ;keine Sekundaeradr.
BA20  20 BA FF  JSR  SETFPAR      ;Fileparameter setzen
BA23  20 C0 FF  JSR  OPEN         ;Datei oeffnen
BA26  A2 01     LDX  #$01
BA28  20 C9 FF  JSR  CHKOUT       ;Datei #1 als Ausgabekan.
BA2B  A9 0D     LDA  #$0D
BA2D  20 D2 FF  JSR  BSOUT        ;Zeilenvorschub ausgeben
BA30  A9 08     LDA  #$08
BA32  20 D2 FF  JSR  BSOUT        ;Code 8 ausgeben
BA35  A9 00     LDA  #$00
BA37  85 5F     STA  $5F          ;akt. Koordinaten
BA39  85 61     STA  $61
BA3B  85 60     STA  $60
BA3D  A9 00     LDA  #$00
BA3F  8D 4E CA  STA  PZEIG        ;Zaehler bis 6 fuer X
BA42  A9 00     LDA  #$00
BA44  8D E2 C5  STA  COPBYTE      ;Sammelbyte
BA47  A9 00     LDA  #$00
BA49  8D E6 C5  STA  COPZAEY      ;Zaehler bis 7 fuer Y

BA4C  18        CLC
BA4D  A5 5F     LDA  $5F          ;$09,$0A = $5F,$60 + PZEIG
BA4F  6D 4E CA  ADC  PZEIG
BA52  85 09     STA  $09
BA54  A5 60     LDA  $60
BA56  69 00     ADC  #$00
BA58  85 0A     STA  $0A
BA5A  18        CLC
BA5B  A5 61     LDA  $61          ;$A4 = $61 + COPZAEY
BA5D  6D E6 C5  ADC  COPZAEY
BA60  85 A4     STA  $A4
BA62  A9 0B     LDA  #$0B
BA64  85 F7     STA  $F7          ;Punktfarbe 11 (TEST)
BA66  20 97 92  JSR  PUNKT        ;Punkt abfragen
```

```
BA69   20 0E 94    JSR   KERROMEIN ;KERNAL einschalten
BA6C   A5 90       LDA   $90       ;Testergebnis
BA6E   F0 10       BEQ   $BA80     ;Punkt frei ?
BA70   C9 08       CMP   #$08      ;Merker f. illegale Koord.
BA72   F0 0C       BEQ   $BA80     ;Punktkoord. illegal ?
BA74   AD E2 C5    LDA   COPBYTE   ;Sammelbyte
BA77   AC E6 C5    LDY   COPZAEY   ;Zaehler in Y-Richtung
BA7A   19 78 97    ORA   MBITTABS,Y ;entspr. Bit setzen
BA7D   8D E2 C5    STA   COPBYTE   ;Byte wieder speichern
BA80   EE E6 C5    INC   COPZAEY   ;Y-Zaehler erh.
BA83   AD E6 C5    LDA   COPZAEY
BA86   C9 07       CMP   #$07      ;7 Hires-Zeilen
BA88   D0 C2       BNE   $BA4C     ;nicht abgel., dann Schl.
BA8A   AD E2 C5    LDA   COPBYTE   ;Sammelbyte
BA8D   09 80       ORA   #$80      ;Bit 7 setzen
BA8F   20 D2 FF    JSR   BSOUT     ;und ausgeben
BA92   EE 4E CA    INC   PZEIG     ;Zaehler f. X erh.
BA95   AD 4E CA    LDA   PZEIG
BA98   C9 06       CMP   #$06      ;6 Hires-Spalten
BA9A   F0 03       BEQ   $BA9F     ;1 Druckzeichen fertig ?
BA9C   4C 42 BA    JMP   $BA42     ;nein, dann Schleife

BA9F   18          CLC
BAA0   A5 5F       LDA   $5F       ;x um 6 erhoehen
BAA2   69 06       ADC   #$06
BAA4   85 5F       STA   $5F
BAA6   A5 60       LDA   $60
BAA8   69 00       ADC   #$00
BAAA   85 60       STA   $60
BAAC   A5 5F       LDA   $5F
BAAE   C9 44       CMP   #$44
BAB0   D0 06       BNE   $BAB8
BAB2   A5 60       LDA   $60
BAB4   C9 01       CMP   #$01
BAB6   F0 03       BEQ   $BABB
BAB8   4C 3D BA    JMP   $BA3D     ;Schleife, bis x = 324

BABB   A9 0D       LDA   #$0D
BABD   20 D2 FF    JSR   BSOUT     ;Zeilenvorschub ausgeben

BAC0   18          CLC
BAC1   A5 61       LDA   $61       ;y um 7 erhoehen
BAC3   69 07       ADC   #$07
BAC5   85 61       STA   $61
BAC7   A9 00       LDA   #$00      ;x = 0
BAC9   85 5F       STA   $5F
BACB   85 60       STA   $60
BACD   A5 61       LDA   $61
BACF   C9 D2       CMP   #$D2      ;194 Hires-Zeilen
BAD1   90 E5       BCC   $BAB8     ;Schleife, bis y gr.= 194
```

```
 BAD3   A9 0D       LDA  #$0D       ;Zeilenvorschub
 BAD5   20 D2 FF    JSR  BSOUT      ;ausgeben
 BAD8   A9 0F       LDA  #$0F       ;Code 15
 BADA   20 D2 FF    JSR  BSOUT      ;ausgeben
 BADD   20 CC FF    JSR  CLRCH      ;Kanaele schliessen
 BAE0   A9 01       LDA  #$01
 BAE2   20 C3 FF    JSR  CLOSE      ;Datei #1 schliessen
 BAE5   20 CC FF    JSR  CLRCH      ;Kanaele schliessen
 BAE8   68          PLA
 BAE9   8D B0 C5    STA  MULTIJN    ;Multi-Flag restaurieren
 BAEC   68          PLA
 BAED   8D 2E C5    STA  XMAXHIGH   ;X-Grenzen restaurieren
 BAF0   68          PLA
 BAF1   8D 2D C5    STA  XMAXLOW
 BAF4   4C 2A 82    JMP  ENDSMB     ;Befehl fertig

BEFREC:
 BAF7   EA          NOP             (???)
 BAF8   20 59 93    JSR  GETXYXYF   ;5 Parameter holen
 BAFB   AD A8 C5    LDA  EY      ;EY,EX,EX+1 nach $61,$5F,$60
 BAFE   85 61       STA  $61     ;(Seiten-Laengen S.L.X,S.L.Y)
 BB00   AD AA C5    LDA  EX
 BB03   85 5F       STA  $5F
 BB05   AD AB C5    LDA  EX+1
 BB08   85 60       STA  $60
 BB0A   A9 00       LDA  #$00
 BB0C   8D A9 C5    STA  EY+1       ;High-Byte von EY auf 0

 BB0F   18          CLC
 BB10   AD 97 C5    LDA  AX         ;EX,EX+1 = AX,AX+1 + S.L.X
 BB13   65 5F       ADC  $5F
 BB15   8D AA C5    STA  EX
 BB18   AD 98 C5    LDA  AX+1
 BB1B   65 60       ADC  $60
 BB1D   8D AB C5    STA  EX+1
 BB20   AD AC C5    LDA  AY
 BB23   8D A8 C5    STA  EY         ;EY = AY
 BB26   20 5E A3    JSR  LINZEI     ;obere Linie zeichnen

 BB29   18          CLC
 BB2A   AD A8 C5    LDA  EY         ;AY = EY
 BB2D   8D AC C5    STA  AY
 BB30   65 61       ADC  $61
 BB32   8D A8 C5    STA  EY         ;EY = AY + S.L.Y
 BB35   AD AA C5    LDA  EX
 BB38   8D 97 C5    STA  AX         ;AX,AX+1 = EX,EX+1
 BB3B   AD AB C5    LDA  EX+1
 BB3E   8D 98 C5    STA  AX+1
 BB41   20 5E A3    JSR  LINZEI     ;rechte Linie zeichnen
```

```
BB44  38        SEC
BB45  AD 97 C5  LDA  AX        ;EX,EX+1 = AX,AX+1 - S.L.X
BB48  E5 5F     SBC  $5F
BB4A  8D AA C5  STA  EX
BB4D  AD 98 C5  LDA  AX+1
BB50  E5 60     SBC  $60
BB52  8D AB C5  STA  EX+1
BB55  AD A8 C5  LDA  EY        ;AY = EY
BB58  8D AC C5  STA  AY
BB5B  20 5E A3  JSR  LINZEI    ;untere Linie zeichnen

BB5E  38        SEC
BB5F  AD A8 C5  LDA  EY        ;EY = EY - S.L.Y
BB62  E5 61     SBC  $61
BB64  8D A8 C5  STA  EY
BB67  AD AA C5  LDA  EX        ;AX,AX+1 = EX,EX+1
BB6A  8D 97 C5  STA  AX
BB6D  AD AB C5  LDA  EX+1
BB70  8D 98 C5  STA  AX+1
BB73  20 5E A3  JSR  LINZEI    ;linke Linie zeichnen
BB76  4C 2A 82  JMP  ENDSMB    ;Befehl fertig

BB79  AD AA C5  LDA  EX        ;Block zeichnen Multi-C.
BB7C  C9 9F     CMP  #$9F
BB7E  90 05     BCC  $BB85     ;EX kleiner 159 ?
BB80  A9 9F     LDA  #$9F      ;sonst EX = 159
BB82  8D AA C5  STA  EX
BB85  AD A8 C5  LDA  EY
BB88  C9 C7     CMP  #$C7
BB8A  90 05     BCC  $BB91     ;EY kleiner 199 ?
BB8C  A9 C7     LDA  #$C7      ;sonst EY = 199
BB8E  8D A8 C5  STA  EY
BB91  A5 09     LDA  $09       ;$6F   = $09
BB93  85 6F     STA  $6F
BB95  29 03     AND  #$03
BB97  0A        ASL  A
BB98  AA        TAX            ;X = 2 * (xk mod 4)
BB99  4C B1 BB  JMP  $BBB1

BB9C  A5 09     LDA  $09       ;X-Koordinate
BB9E  29 03     AND  #$03
BBA0  0A        ASL  A
BBA1  AA        TAX            ;X = 2 * (xk mod 4)
BBA2  D0 0D     BNE  $BBB1     ;X ungleich 0 ?
BBA4  18        CLC
BBA5  A5 A8     LDA  $A8       ;$A8,$A9 = $A8,$A9 + 8
BBA7  69 08     ADC  #$08
BBA9  85 A8     STA  $A8
BBAB  A5 A9     LDA  $A9
BBAD  69 00     ADC  #$00
BBAF  85 A9     STA  $A9
```

```
BBB1  A5 A4      LDA  $A4        ;Y-Koord.
BBB3  29 07      AND  #$07
BBB5  A8         TAY             ;(YK. and 7) nach Y

BBB6  A5 F7      LDA  $F7        ;Punktfarbe
BBB8  C9 02      CMP  #$02
BBBA  D0 15      BNE  $BBD1      ;nicht 2 ?
BBBC  20 06 94   JSR  KERROMAUS ;Kernal aus
BBBF  B1 A8      LDA  ($A8),Y
BBC1  1D 30 93   ORA  GBITTABS,X ;1.Bit setzen
BBC4  91 A8      STA  ($A8),Y
BBC6  E8         INX
BBC7  B1 A8      LDA  ($A8),Y
BBC9  3D 40 93   AND  GBITTABL,X ;2.Bit loeschen
BBCC  91 A8      STA  ($A8),Y
BBCE  4C 2E BC   JMP  $BC2E

BBD1  C9 00      CMP  #$00
BBD3  D0 15      BNE  $BBEA       ;Punktfarbe nicht 0 ?
BBD5  20 06 94   JSR  KERROMAUS
BBD8  B1 A8      LDA  ($A8),Y
BBDA  3D 40 93   AND  GBITTABL,X ;1.Bit loeschen
BBDD  91 A8      STA  ($A8),Y
BBDF  E8         INX
BBE0  B1 A8      LDA  ($A8),Y
BBE2  3D 40 93   AND  GBITTABL,X ;2.Bit loeschen
BBE5  91 A8      STA  ($A8),Y
BBE7  4C 2E BC   JMP  $BC2E

BBEA  C9 01      CMP  #$01
BBEC  D0 15      BNE  $BC03       ;Punktfarbe nicht 1 ?
BBEE  20 06 94   JSR  KERROMAUS
BBF1  B1 A8      LDA  ($A8),Y
BBF3  3D 40 93   AND  GBITTABL,X ;1.Bit loeschen
BBF6  91 A8      STA  ($A8),Y
BBF8  E8         INX
BBF9  B1 A8      LDA  ($A8),Y
BBFB  1D 30 93   ORA  GBITTABS,X ;2.Bit setzen
BBFE  91 A8      STA  ($A8),Y
BC00  4C 2E BC   JMP  $BC2E

BC03  C9 04      CMP  #$04
BC05  F0 15      BEQ  $BC1C        ;Punktfarbe = 4 ?
BC07  20 06 94   JSR  KERROMAUS
BC0A  B1 A8      LDA  ($A8),Y
BC0C  1D 30 93   ORA  GBITTABS,X ;1.Bit setzen
BC0F  91 A8      STA  ($A8),Y
BC11  E8         INX
BC12  B1 A8      LDA  ($A8),Y
BC14  1D 30 93   ORA  GBITTABS,X ;2.Bit setzen
BC17  91 A8      STA  ($A8),Y
```

```
 BC19   4C 2E BC   JMP  $BC2E

 BC1C   20 06 94   JSR  KERROMAUS
 BC1F   B1 A8      LDA  ($A8),Y
 BC21   5D 30 93   EOR  GBITTABS,X ;1. Bit invertieren
 BC24   91 A8      STA  ($A8),Y
 BC26   E8         INX
 BC27   B1 A8      LDA  ($A8),Y
 BC29   5D 30 93   EOR  GBITTABS,X ;2. Bit invertieren
 BC2C   91 A8      STA  ($A8),Y

 BC2E   E6 09      INC  $09        ;X erhoehen
 BC30   A5 09      LDA  $09
 BC32   C9 A0      CMP  #$A0
 BC34   B0 07      BCS  $BC3D      ;X groesser gleich 160 ?
 BC36   CD AA C5   CMP  EX
 BC39   90 30      BCC  $BC6B      ;X kleiner EX ?
 BC3B   F0 2E      BEQ  $BC6B      ;X gleich EX ?

 BC3D   E6 A4      INC  $A4        ;Y erhoehen
 BC3F   A5 A4      LDA  $A4
 BC41   29 07      AND  #$07
 BC43   D0 0D      BNE  $BC52      ;(Y mod 8) ungleich 0 ?
 BC45   18         CLC
 BC46   A5 AA      LDA  $AA        ;$AA,$AB = $AA,$AB + 320
 BC48   69 40      ADC  #$40
 BC4A   85 AA      STA  $AA
 BC4C   A5 AB      LDA  $AB
 BC4E   69 01      ADC  #$01
 BC50   85 AB      STA  $AB
 BC52   A5 AA      LDA  $AA        ;$A8,$A9 = $AA,$AB
 BC54   85 A8      STA  $A8
 BC56   A5 AB      LDA  $AB
 BC58   85 A9      STA  $A9
 BC5A   A5 6F      LDA  $6F        ;$09   = $6F
 BC5C   85 09      STA  $09
 BC5E   A5 A4      LDA  $A4        ;Y-Koord.
 BC60   CD A8 C5   CMP  EY
 BC63   F0 03      BEQ  $BC68      ;YK gleich EY ?
 BC65   90 01      BCC  $BC68      ;YK kleiner EY ?
 BC67   60         RTS             ;fertig mit Block

 BC68   4C 79 BB   JMP  $BB79      ;zur (aeusseren) Schleife

 BC6B   4C 9C BB   JMP  $BB9C      ;zur (inneren) Schleife

BEFBLOCK:
 BC6E   20 59 93   JSR  GETXYXYF ;AX,AY,EX,EY und Pktf.
 BC71   20 77 BC   JSR  BLOCKZEI ;Block zeichnen
 BC74   4C 2A 82   JMP  ENDSMB   ;Befehl fertig
```

```
BLOCKZEI:
 BC77  20 06 94   JSR  KERROMAUS ;Kernal aus
 BC7A  AE AC C5   LDX  AY         ;AX mit AY vertauschen
 BC7D  AD 97 C5   LDA  AX
 BC80  8D AC C5   STA  AY
 BC83  8E 97 C5   STX  AX
 BC86  AE AD C5   LDX  AY+1
 BC89  AD 98 C5   LDA  AX+1
 BC8C  8D AD C5   STA  AY+1
 BC8F  8E 98 C5   STX  AX+1
 BC92  AD AD C5   LDA  AY+1  ;Anfangswerte als akt. Koord.
 BC95  85 0A      STA  $0A
 BC97  AD 98 C5   LDA  AX+1
 BC9A  85 A5      STA  $A5
 BC9C  AD AC C5   LDA  AY
 BC9F  85 09      STA  $09
 BCA1  AD 97 C5   LDA  AX
 BCA4  85 A4      STA  $A4
 BCA6  A9 FF      LDA  #$FF
 BCA8  8D AE C5   STA  GFLAG ;Flag fuer Pkt. zeichnen j/n
 BCAB  20 97 92   JSR  PUNKT ;Koordinaten umrechnen
 BCAE  A9 00      LDA  #$00
 BCB0  8D AE C5   STA  GFLAG ;Flag ruecksetzen
 BCB3  A5 90      LDA  $90   ;Status
 BCB5  C9 08      CMP  #$08
 BCB7  D0 05      BNE  $BCBE ;kein illegaler Bereich ?
 BCB9  A9 00      LDA  #$00    ;Nr. f. 'bad mode'
 BCBB  4C 8C 88   JMP  SERROUT ;Fehler ausgeben

 BCBE  38         SEC
 BCBF  A5 A8      LDA  $A8     ;$A8,$A9 = $A8,$A9 - 99
 BCC1  E5 63      SBC  $63
 BCC3  85 A8      STA  $A8
 BCC5  A5 A9      LDA  $A9
 BCC7  E9 00      SBC  #$00
 BCC9  85 A9      STA  $A9
 BCCB  18         CLC
 BCCC  AD AC C5   LDA  AY    ;Anfangswerte als akt. Koord.
 BCCF  85 09      STA  $09
 BCD1  AD AD C5   LDA  AY+1
 BCD4  85 0A      STA  $0A
 BCD6  AD 97 C5   LDA  AX
 BCD9  85 A4      STA  $A4
 BCDB  A5 A8      LDA  $A8   ;$AA,$AB = $A8,$A9
 BCDD  85 AA      STA  $AA
 BCDF  A5 A9      LDA  $A9
 BCE1  85 AB      STA  $AB
 BCE3  AD B0 C5   LDA  MULTIJN ;Multi-Colour-Flag
 BCE6  C9 2C      CMP  #$2C
 BCE8  D0 03      BNE  $BCED   ;kein Multi-Colour ?
 BCEA  4C 79 BB   JMP  $BB79   ;zum Multi-C.-Block
```

```
BCED   A5 09       LDA  $09          ;$6F,$70 = $09,$0A
BCEF   85 6F       STA  $6F
BCF1   A5 0A       LDA  $0A
BCF3   85 70       STA  $70
BCF5   A5 09       LDA  $09
BCF7   29 07       AND  #$07
BCF9   AA          TAX               ;X = (xk mod 8)
BCFA   4C 11 BD    JMP  $BD11

BCFD   A5 09       LDA  $09
BCFF   29 07       AND  #$07
BD01   AA          TAX               ;X = (xk mod 8)
BD02   D0 0D       BNE  $BD11
BD04   18          CLC
BD05   A5 A8       LDA  $A8          ;$A8,$A9 = $A8,$A9 + 8
BD07   69 08       ADC  #$08
BD09   85 A8       STA  $A8
BD0B   A5 A9       LDA  $A9
BD0D   69 00       ADC  #$00
BD0F   85 A9       STA  $A9

BD11   A5 A4       LDA  $A4
BD13   29 07       AND  #$07
BD15   A8          TAY               ;Y = (yk mod 8)

BD16   A5 F7       LDA  $F7          ;Punktfarbe
BD18   C9 02       CMP  #$02
BD1A   D0 0D       BNE  $BD29        ;nicht 2 ?
BD1C   20 06 94    JSR  KERROMAUS
BD1F   B1 A8       LDA  ($A8),Y
BD21   5D 30 93    EOR  GBITTABS,X ;Bit invertieren
BD24   91 A8       STA  ($A8),Y
BD26   4C 44 BD    JMP  $BD44

BD29   C9 00       CMP  #$00
BD2B   D0 0D       BNE  $BD3A         ;Farbe nicht 0 ?
BD2D   20 06 94    JSR  KERROMAUS
BD30   B1 A8       LDA  ($A8),Y
BD32   3D 40 93    AND  GBITTABL,X ;Bit loeschen
BD35   91 A8       STA  ($A8),Y
BD37   4C 44 BD    JMP  $BD44

BD3A   20 06 94    JSR  KERROMAUS
BD3D   B1 A8       LDA  ($A8),Y
BD3F   1D 30 93    ORA  GBITTABS,X ;Bit setzen
BD42   91 A8       STA  ($A8),Y

BD44   E6 09       INC  $09         ;$09,$0A erhoehen
BD46   D0 02       BNE  $BD4A
BD48   E6 0A       INC  $0A
BD4A   A5 0A       LDA  $0A
```

```
BD4C   F0 09      BEQ  $BD57    ;xk kleiner 256 ?
BD4E   A5 09      LDA  $09
BD50   C9 40      CMP  #$40
BD52   90 03      BCC  $BD57    ;xk kleiner 320 ?
BD54   4C 74 BD   JMP  $BD74    ;sonst zu $BD74
BD57   A5 0A      LDA  $0A      ;$09,$0A mit EX,EX+1 vgl.
BD59   8D 24 C5   STA  VGLA+1
BD5C   A5 09      LDA  $09
BD5E   8D 23 C5   STA  VGLA
BD61   AD AA C5   LDA  EX
BD64   8D 1A C5   STA  VGLE
BD67   AD AB C5   LDA  EX+1
BD6A   8D 1B C5   STA  VGLE+1
BD6D   20 16 A6   JSR  VGLAE    ;Vergleichsroutine
BD70   F0 8B      BEQ  $BCFD    ;gleich, dann Schleife
BD72   90 89      BCC  $BCFD    ;kleiner, dann Schleife

BD74   E6 A4      INC  $A4      ;yk erhoehen
BD76   A5 A4      LDA  $A4
BD78   29 07      AND  #$07
BD7A   D0 0D      BNE  $BD89    ;(yk mod 8) ungleich 0 ?
BD7C   18         CLC
BD7D   A5 AA      LDA  $AA      ;$AA,$AB = $AA,$AB + 320
BD7F   69 40      ADC  #$40
BD81   85 AA      STA  $AA
BD83   A5 AB      LDA  $AB
BD85   69 01      ADC  #$01
BD87   85 AB      STA  $AB

BD89   A5 AA      LDA  $AA      ;$A8,$A9 = $AA,$AB
BD8B   85 A8      STA  $A8
BD8D   A5 AB      LDA  $AB
BD8F   85 A9      STA  $A9
BD91   A5 6F      LDA  $6F      ;$09,$0A = $6F,$70
BD93   85 09      STA  $09
BD95   A5 70      LDA  $70
BD97   85 0A      STA  $0A
BD99   A5 A4      LDA  $A4
BD9B   C9 C8      CMP  #$C8
BD9D   B0 07      BCS  $BDA6    ;yk groesser= 200 ?
BD9F   CD A8 C5   CMP  EY
BDA2   F0 03      BEQ  $BDA7    ;yk gleich EY ?
BDA4   90 01      BCC  $BDA7    ;yk kleiner EY ?
BDA6   60         RTS           ;Block fertig

BDA7   4C ED BC   JMP  $BCED    ;zur (aeusseren) Schleife
```

```
DSCBMOUT1:
 BDAA   4C 5A 9A    JMP   DSCBMOUT ;zur Ausgabe von 'ds-cbm'

BEFMEM:
 BDAD   A9 CC       LDA   #$CC
 BDAF   8D 88 02    STA   VIDRAMHI ;Video-RAM nach $CC00
 BDB2   A9 93       LDA   #$93
 BDB4   20 D2 FF    JSR   BSOUT     ;Bildschirm loeschen
 BDB7   A9 00       LDA   #$00      ;$20,$21 auf $D000
 BDB9   85 20       STA   $20
 BDBB   A9 D0       LDA   #$D0
 BDBD   85 21       STA   $21
 BDBF   78          SEI            ;Interrupts verhindern
 BDC0   A5 01       LDA   $01      ;Prozessor-Port
 BDC2   29 FB       AND   #$FB     ;Zeichengenerator lesbar
 BDC4   85 01       STA   $01
 BDC6   A9 00       LDA   #$00     ;$A8,$A9 auf $E000
 BDC8   85 A8       STA   $A8
 BDCA   A9 E0       LDA   #$E0
 BDCC   85 A9       STA   $A9
 BDCE   A0 00       LDY   #$00
 BDD0   A2 00       LDX   #$00
 BDD2   B1 20       LDA   ($20),Y ;Byte aus Zeichengen.-ROM
 BDD4   91 A8       STA   ($A8),Y ;ab $E000 ablegen
 BDD6   E6 20       INC   $20      ;$20,$21 erhoehen
 BDD8   D0 03       BNE   $BDDD
 BDDA   E6 21       INC   $21
 BDDC   E8          INX            ;Zaehler (high)
 BDDD   20 F0 9A    JSR   INCA8A9 ;$A8,$A9 erhoehen
 BDE0   E0 10       CPX   #$10
 BDE2   D0 EE       BNE   $BDD2    ;X ungl. 16, dann Schleife
 BDE4   A5 01       LDA   $01      ;Prozessor-Port
 BDE6   09 04       ORA   #$04     ;Zeich.gen. unlesbar
 BDE8   85 01       STA   $01
 BDEA   A9 94       LDA   #$94
 BDEC   8D 00 DD    STA   CIA2PRA ;VIC-Bereich ab $C000
 BDEF   A9 39       LDA   #$39     ;Video-Ram ab $CC00 und
 BDF1   8D 18 D0    STA   VICADS   ;Zeichengen. ab $E000
 BDF4   4C 74 91    JMP   BEF0     ;zum Leerbefehl

BEFDETECT:
 BDF7   20 02 82    JSR   SGETBYTN   ;Typ holen
 BDFA   8E 15 C5    STX   DETECTART ;merken
 BDFD   BD 1E D0    LDA   $D01E,X    ;entspr. VIC-Register
 BE00   8D 14 C5    STA   DETECTERG ;als Ergebnis merken
 BE03   4C 2A 82    JMP   ENDSMB     ;Befehl fertig

TEXTKEY:
 BE06   4B 45 59 00                  ;"KEY"
```

```
TEXTCHR:
 BE0A   43 48 52 24 28 31 33 29 00   ;"CHR$(13)"

BEFDISPLAY:
 BE13   20 DB 83    JSR   INCBASBZ ;Code ueberlesen
 BE16   A2 00       LDX   #$00      ;$A6,$A7 = $0001
 BE18   86 A7       STX   $A7
 BE1A   A9 01       LDA   #$01
 BE1C   85 A6       STA   $A6
 BE1E   A9 C6       LDA   #$C6      ;$20,$21 auf KEYTAB
 BE20   85 21       STA   $21
 BE22   A9 4D       LDA   #$4D
 BE24   85 20       STA   $20
 BE26   A0 00       LDY   #$00
 BE28   B9 06 BE    LDA   TEXTKEY,Y ;Text "KEY" ausgeben
 BE2B   F0 06       BEQ   $BE33     ;kein Endekriterium ?
 BE2D   20 D2 FF    JSR   BSOUT     ;Zeichen ausgeben
 BE30   C8          INY
 BE31   D0 F5       BNE   $BE28     ;Schleife
 BE33   A5 A6       LDA   $A6       ;Key-Nr. nach $63,$62
 BE35   85 63       STA   $63
 BE37   A5 A7       LDA   $A7
 BE39   85 62       STA   $62
 BE3B   A2 90       LDX   #$90
 BE3D   38          SEC
 BE3E   20 88 80    JSR   SXFLP     ;nach Fliessk. wandeln
 BE41   20 B5 80    JSR   SFACASC   ;in Text wandeln
 BE44   20 2B A0    JSR   OUTASC    ;Text ausgeben
 BE47   A9 2C       LDA   #$2C      ;","
 BE49   20 D2 FF    JSR   BSOUT     ;Komma ausgeben
 BE4C   A9 22       LDA   #$22      ;"
 BE4E   20 D2 FF    JSR   BSOUT     ;Anf.zeich. ausgeben
 BE51   A9 01       LDA   #$01
 BE53   8D AC C5    STA   AY        ;Flag f. " am Ende
 BE56   A2 00       LDX   #$00
 BE58   A0 00       LDY   #$00
 BE5A   B1 20       LDA   ($20),Y   ;Zeichen aus KEYTAB
 BE5C   F0 22       BEQ   $BE80     ;kein Endezeichen ?
 BE5E   C9 0D       CMP   #$0D
 BE60   F0 45       BEQ   $BEA7     ;chr$(13) ?
 BE62   AD AC C5    LDA   AY
 BE65   D0 0F       BNE   $BE76
 BE67   A9 2C       LDA   #$2C
 BE69   20 D2 FF    JSR   BSOUT     ;Komma ausgeben
 BE6C   A9 22       LDA   #$22
 BE6E   20 D2 FF    JSR   BSOUT     ;Anf.zeichen ausgeben
 BE71   A9 01       LDA   #$01
 BE73   8D AC C5    STA   AY
 BE76   B1 20       LDA   ($20),Y   ;Zeichen aus KEYTAB
 BE78   20 D2 FF    JSR   BSOUT     ;ausgeben
 BE7B   E8          INX
```

```
BE7C  C8          INY
BE7D  4C 5A BE    JMP   $BE5A       ;Schleife

BE80  E6 A6       INC   $A6         ;akt. Keynr. erhoehen
BE82  18          CLC
BE83  A5 20       LDA   $20         ;$20,$21 = $20,$21 + 16
BE85  69 10       ADC   #$10
BE87  85 20       STA   $20
BE89  A5 21       LDA   $21
BE8B  69 00       ADC   #$00
BE8D  85 21       STA   $21
BE8F  AD AC C5    LDA   AY          ;"-Flag
BE92  F0 05       BEQ   $BE99       ;nicht gesetzt ?
BE94  A9 22       LDA   #$22
BE96  20 D2 FF    JSR   BSOUT       ;Anf.Zeich. ausgeben
BE99  A9 0D       LDA   #$0D
BE9B  20 D2 FF    JSR   BSOUT       ;(cr) ausgeben
BE9E  A5 A6       LDA   $A6         ;Keynr.
BEA0  C9 11       CMP   #$11
BEA2  D0 82       BNE   $BE26       ;nicht gleich 17 ?
BEA4  4C 2A 82    JMP   ENDSMB      ;Befehl fertig

BEA7  AD AC C5    LDA   AY          ;"-Flag
BEAA  F0 0F       BEQ   $BEBB       ;nicht gesetzt ?
BEAC  A9 22       LDA   #$22
BEAE  20 D2 FF    JSR   BSOUT       ;Anf.zeichen ausgeben
BEB1  A9 2B       LDA   #$2B        ;"+"
BEB3  20 D2 FF    JSR   BSOUT       ;Pluszeichen ausgeben
BEB6  A9 00       LDA   #$00
BEB8  8D AC C5    STA   AY          ;"-Flag ruecksetzen
BEBB  A2 00       LDX   #$00
BEBD  BD 0A BE    LDA   TEXTCHR,X  ;"CHR$(13)"
BEC0  F0 06       BEQ   $BEC8       ;Endemarkierung ?
BEC2  20 D2 FF    JSR   BSOUT       ;Zeichen ausgeben
BEC5  E8          INX
BEC6  D0 F5       BNE   $BEBD       ;Schleife
BEC8  4C 7B BE    JMP   $BE7B       ;weiter wie oben

SLIST3:
BECB  08          PHP               ;Status retten
BECC  8C FA 03    STY   LYMERK      ;X und Y merken
BECF  8E FB 03    STX   LXMERK
BED2  AC 8D 02    LDY   SHCOCTFL ;Shift/C=/CTRL - Flag
BED5  C0 01       CPY   #$01
BED7  D0 0E       BNE   $BEE7       ;kein Shift ?
BED9  A2 00       LDX   #$00
BEDB  C8          INY
BEDC  D0 FD       BNE   $BEDB       ;kleine Warteschleife
BEDE  E8          INX
BEDF  EC F8 C5    CPX   DELAY       ;Delay-Parameter
BEE2  D0 F7       BNE   $BEDB       ;grosse Warteschleife
```

```
BEE4  4C F2 BE   JMP  $BEF2     ;weiter

BEE7  AC 8D 02   LDY  SHCOCTFL ;Shift/C=/CTRL - Flag
BEEA  C0 04      CPY  #$04
BEEC  F0 04      BEQ  $BEF2     ;CTRL-Taste ?
BEEE  C0 00      CPY  #$00
BEF0  D0 F5      BNE  $BEE7     ;irgendeine dieser Tasten?
BEF2  AC 8C C5   LDY  PAGEFLAG
BEF5  C0 0A      CPY  #$0A
BEF7  D0 44      BNE  $BF3D     ;PAGE nicht aktiv ?
BEF9  AC 8D C5   LDY  PAGEWERT
BEFC  C4 D6      CPY  $D6       ;Cursorzeile
BEFE  B0 3D      BCS  $BF3D     ;Pagewert groesser= Z. ?
BF00  A0 00      LDY  #$00
BF02  48         PHA
BF03  B1 D1      LDA  ($D1),Y   ;akt. BS-Zeile
BF05  99 44 02   STA  $0244,Y   ;ab $0244 ablegen
BF08  C9 20      CMP  #$20      ;Leerzeichen
BF0A  F0 04      BEQ  $BF10     ;ja ?
BF0C  C8         INY            ;naechstes Zeichen
BF0D  4C 03 BF   JMP  $BF03     ;Schleife

BF10  EA         NOP            ;(???)
BF11  20 E4 FF   JSR  GETIN     ;Zeichen von Tastatur
BF14  C9 0D      CMP  #$0D
BF16  D0 F9      BNE  $BF11     ;nicht (RETURN) ?
BF18  A5 D4      LDA  $D4       ;"-Flag retten
BF1A  48         PHA
BF1B  A9 00      LDA  #$00
BF1D  85 D4      STA  $D4       ;"-Flag = 0
BF1F  A9 93      LDA  #$93      ;(clr)
BF21  20 D2 FF   JSR  BSOUT     ;Bildschirm loeschen
BF24  A0 00      LDY  #$00
BF26  A9 20      LDA  #$20      ;" "
BF28  20 D2 FF   JSR  BSOUT     ;ausgeben
BF2B  B9 44 02   LDA  $0244,Y   ;gespeicherte Zeile
BF2E  C9 20      CMP  #$20      ;Leerzeichen
BF30  F0 07      BEQ  $BF39     ;ja ?
BF32  99 00 04   STA  $0400,Y   ;in Video-RAM ablegen
BF35  C8         INY
BF36  4C 26 BF   JMP  $BF26     ;Schleife

BF39  68         PLA
BF3A  85 D4      STA  $D4       ;"-Flag restaurieren
BF3C  68         PLA
BF3D  A2 83      LDX  #$83      ;$20,$21 auf $83E2
BF3F  86 21      STX  $21       ;(Befehlstabelle)
BF41  A2 E2      LDX  #$E2
BF43  86 20      STX  $20
BF45  C9 64      CMP  #$64      ;Code = $64 (Simon-Prefix)
BF47  F0 09      BEQ  $BF52     ;ja ?
```

```
BF49  AC FA 03   LDY  LYMERK    ;X und Y restaurieren
BF4C  AE FB 03   LDX  LXMERK
BF4F  4C 4A 82   JMP  SLIST1    ;Basic-List f. 1 Befehl

BF52  A6 D4      LDX  $D4       ;"-Flag
BF54  E0 00      CPX  #$00
BF56  D0 F1      BNE  $BF49     ;"-Modus aktiv ?
BF58  A0 00      LDY  #$00
BF5A  84 22      STY  $22       ;Zaehler auf 0
BF5C  AC FA 03   LDY  LYMERK    ;altes Y holen
BF5F  C8         INY            ;erhoehen
BF60  B1 5F      LDA  ($5F),Y   ;Zeiger in Programm
BF62  AA         TAX            ;Code nach X
BF63  CA         DEX
BF64  8A         TXA            ;X-1 nach Akku
BF65  A0 00      LDY  #$00
BF67  8D D5 C5   STA  LCODE     ;Code merken
BF6A  48         PHA            ;und auf Stapel
BF6B  AD DC C5   LDA  OPTFLAG   ;Option-Flag
BF6E  C9 0A      CMP  #$0A
BF70  D0 05      BNE  $BF77     ;Option nicht aktiv ?
BF72  A9 12      LDA  #$12
BF74  20 D2 FF   JSR  BSOUT     ;(RVS on) ausgeben
BF77  68         PLA            ;Code holen
BF78  C9 00      CMP  #$00
BF7A  F0 21      BEQ  $BF9D     ;1. Befehl ?
BF7C  B1 20      LDA  ($20),Y   ;Zeichen aus TABBEF
BF7E  C9 40      CMP  #$40      ;Trennzeichen (Klammeraffe)
BF80  F0 09      BEQ  $BF8B     ;ja ?
BF82  E6 20      INC  $20       ;$20,$21 erhoehen
BF84  D0 02      BNE  $BF88
BF86  E6 21      INC  $21
BF88  4C 7C BF   JMP  $BF7C     ;Schleife

BF8B  E6 22      INC  $22       ;Zaehler erhoehen
BF8D  A5 22      LDA  $22
BF8F  CD D5 C5   CMP  LCODE
BF92  F0 03      BEQ  $BF97     ;gleich Code ?
BF94  4C 82 BF   JMP  $BF82     ;Schleife

BF97  E6 20      INC  $20       ;$20,$21 erhoehen
BF99  D0 02      BNE  $BF9D
BF9B  E6 21      INC  $21
BF9D  B1 20      LDA  ($20),Y   ;Befehlswort
BF9F  C9 40      CMP  #$40      ;Endezeichen
BFA1  F0 07      BEQ  $BFAA     ;ja ?
BFA3  20 D2 FF   JSR  BSOUT     ;Zeichen ausgeben
BFA6  C8         INY            ;naechstes Zeichen
BFA7  4C 9D BF   JMP  $BF9D     ;Schleife
```

```
 BFAA  A9 92       LDA  #$92      ;(RVS off)
 BFAC  20 D2 FF    JSR  BSOUT     ;ausgeben
 BFAF  AC FA 03    LDY  LYMERK    ;Y und X herstellen
 BFB2  C8          INY
 BFB3  AE FB 03    LDX  LXMERK
 BFB6  A9 02       LDA  #$02     ;(sinnloser Code)
 BFB8  4C 53 82    JMP  SLIST2   ;zum Basic-List

ENDE:
 BFBB  78          SEI           ;Code wahrscheinlich zur
                                 ;Kennzeichnung der Version

;*** nach $BFBB folgen einige Bytes, die jedoch keine
;*** Bedeutung haben und deshalb auch nicht abgebildet
;*** wurden.
```

6.3 Tabellen der Variablen und Labels (Marken)

Die in der Spalte unter 'T' angegebenen Abkürzungen haben folgende Bedeutung:

A - Tabelle von konstanten Adressen (z.B. Sprungtab.)
b - Simon's-Befehl, -Funktion oder wichtige Aktion
B - Basic-Routine
c - Konstante
C - Tabelle von Konstanten
K - KERNAL-Routine
p - Peripherie-Register (VIC,CIA,SID)
r - (Hilfs-)Routine von Simon's Basic
s - Systemvariable
S - Sytemvektor
T - Textkonstante
v - Variable / Hilfszelle für Simon's Basic
V - Feld von Variablen für Simon's Basic
w - 2-Byte-Variable für Simon's Basic
W - Feld von 2-Byte-Variablen für Simon's Basic

Symbolname	Hex.	T	Bedeutung
20PL40	94CE	r	erhöht $20,$21 um 40
23PL40	94DC	r	erhöht $23,$24 um 40
A4MAL40	945F	r	multipliziert $A4,$A5 mit 40
A8A9BOGM	AA3D	r	wandelt Winkel $A8,A9 in Bogm.
A8A9DIV90	AAD7	r	ganzz. Div. von $A8,$A9 / 90
ADTAB	CBA4	V	Tabelle f. Attack/Decay-Werte
AKTFARB	0286	s	Aktuelle Farbe
ARC1	A88D	r	zeichnet Polygon
AUTOFLAG	C5B2	v	AUTO aktiv, dann =10; sonst 0
AUTOINC	C5B1	v	Inkrement für AUTO-Befehl
AUTOZI	01FE	w	Zähler für AUTO-Befehl
AX	C597	w	Anfangswert X f. Linie (u.a.)
AY	C5AC	w	Anfangswert Y f. Linie (u.a.)
BADMODE	888A	b	Fehlermeldung 'bad mode'
BASBZK	C5F0	w	Programmzählerspeich. f. ONKEY
BASIN	FFCF	K	Zeichen von Eing.ger. holen
BASROMAU	82EC	r	Basic-ROM ausblenden (LORAM=0)
BASROMEI	82F3	r	Basic-ROM einschalten (LORAM=1)
BEFO	9174	b	Leerbefehl (z.B. DISAPA)
BEFANGL	AB10	b	ANGL-Befehl
BEFARC	A876	b	ARC-Befehl
BEFAUTO	9BD6	b	AUTO-Befehl

Symbolname	Hex.		Bedeutung
BEFBCKGNDS	A7B5	b	BCKGNDS-Befehl
BEFBFLASH	9669	b	FLASH-Befehl
BEFBLOCK	BC6E	b	BLOCK-Befehl
BEFCALL	9C2A	b	CALL-Befehl
BEFCENTRE	9947	b	CENTRE-Befehl
BEFCGOTO	99E8	b	CGOTO-Befehl
BEFCHAR	A186	b	CHAR-Befehl
BEFCIRCLE	9489	b	CIRCLE-Befehl
BEFCMOB	A7A6	b	CMOB-Befehl
BEFCOLD	8147	b	COLD-Befehl
BEFCOLOUR	9537	b	COLOUR-Befehl
BEFCOPY	B9F1	b	COPY-Befehl
BEFCSET	B30D	b	CSET-Befehl
BEFDELAY	9E42	b	DELAY-Befehl
BEFDESIGN	A62E	b	DESIGN-Befehl
BEFDETECT	BDF7	b	DETECT-Befehl (auch CHECK-Bef.)
BEFDIR	9546	b	DIR-Befehl
BEFDISABLE	9DB2	b	DISABLE-Befehl
BEFDISAPA	9174	b	DISAPA-Befehl
BEFDISK	B35D	b	DISK-Befehl
BEFDISPLAY	BE13	b	DISPLAY-Befehl
BEFDRAW	A057	b	DRAW-Befehl
BEFDUMP	9F3F	b	DUMP-Befehl
BEFENDLOOP	9D6F	b	END LOOP-Befehl
BEFENDPROC	9D19	b	END PROC-Befehl
BEFENVELOPE	9979	b	ENVELOPE-Befehl
BEFEXEC	9CE3	b	EXEC-Befehl
BEFEXIT	9D37	b	EXIT-Befehl (EXIT IF)
BEFFCHR	A2D0	b	FCHR-Befehl
BEFFCOL	A2E2	b	FCOL-Befehl
BEFFETCH	B004	b	FETCH-Befehl
BEFFILL	A297	b	FILL-Befehl
BEFFIND	A594	b	FIND-Befehl
BEFFLASH	9790	b	FLASH-Befehl
BEFGLOBAL	A33C	b	GLOBAL-Befehl
BEFHICOL	A270	b	HI COL-Befehl
BEFHIRES	91FF	b	HIRES-Befehl
BEFHRDCPY	B4A0	b	HRDCOPY-Befehl
BEFIF	9B6C	b	IF-Befehl (Simon's Version)
BEFKEY	B590	b	KEY-Befehl
BEFLINE	9350	b	LINE-Befehl
BEFLOCAL	A2F4	b	LOCAL-Befehl
BEFLOOP	9E24	b	LOOP-Befehl
BEFLOWCOL	93DC	b	LOW COL-Befehl
BEFMEM	BDAD	b	MEM-Befehl
BEFMERGE	8344	b	MERGE-Befehl
BEFMMOB	95C8	b	MMOB-Befehl

Symbolname	Hex.		Bedeutung
BEFMOBOFF	A856	b	MOB OFF-Befehl
BEFMOBSET	96DA	b	MOB SET-Befehl
BEFMULTI	94EA	b	MULTI-Befehl
BEFMUSIC	9759	b	MUSIC-Befehl
BEFNOERR	9E94	b	NO ERROR-Befehl
BEFNRM	A831	b	NRM-Befehl
BEFOFF	A865	b	OFF-Befehl
BEFOLD	9EDA	b	OLD-Befehl
BEFONERR	9E6E	b	ON ERR-Befehl
BEFONKEY	9D89	b	ON KEY-Befehl
BEFOPTION	9B63	b	OPTION-Befehl
BEFOUT	9ED1	b	OUT-Befehl
BEFPAGE	9836	b	PAGE-Befehl
BEFPAINT	B5E8	b	PAINT-Befehl
BEFPAUSE	A7E2	b	PAUSE-befehl
BEFPLAY	9918	b	PLAY-Befehl (IRQ-R. bei $8A61)
BEFPLOT	9267	b	PLOT-Befehl
BEFPROC	9F30	b	PROC-Befehl
BEFRCOMP	9F1F	b	RCOMP-Befehl
BEFREC	BAF7	b	REC-Befehl
BEFRENUM	837F	b	RENUMBER-Befehl
BEFREPEAT	9AF7	b	REPEAT-Befehl
BEFRESET	9C05	b	RESET-Befehl
BEFRESUME	9E0F	b	RESUME-Befehl
BEFRETRACE	9B48	b	RETRACE-Befehl
BEFRLOCM	A768	b	RLOCMOB-Befehl
BEFROT	A15E	b	ROT-Befehl
BEFSCRLD	B1DC	b	SCRLD-Befehl
BEFSCRSV	B18F	b	SCRSV-Befehl
BEFSECURE	9E4B	b	SECURE-Befehl
BEFTEXT	B229	b	TEXT-Befehl
BEFTRACE	9B5A	b	TRACE-Befehl
BEFUNTIL	9B15	b	UNTIL-Befehl
BEFUSE	B393	b	USE-Befehl
BEFVOL	B33F	b	VOL-Befehl
BEFWAVE	99F7	b	WAVE-Befehl
BFLASHF1	C5D9	v	1. Farbe für BFLASH
BFLASHF2	C5DA	v	2. Farbe für BFLASH
BFLASHJN	C5DB	v	BFLASH-Flag (10=aktiv)
BFLASHSP	C5D7	v	BFLASH-Geschwindigkeit
BFLZAE	C5D8	v	IRQ-Zähler für BFLASH
BINCON	9A09	r	Wandelt Binäre Zahl um
BIRQ	EA31	B	(Normale) Basic-IRQ-Routine
BITWTAB	9788	C	Tabelle der Bitwertigkeiten
BLOCKZEI	BC77	r	Block zeichnen
BNMIVEC	A002	S	NMI-Vektor, wenn Basic aktiv
BSOUT	FFD2	K	ein Zeichen ausgeben

Symbolname	Hex.	T	Bedeutung
BWFLAG	CB82	v	Flag für rollen mit B bzw. W
CHARENO	A04C	r	Zeich.gen. lesbar machen
CHAREN1	A045	r	Zeich.gen. unlesbar machen
CHARZEI	A1B4	r	ein Zeich. im Hires-Modus malen
CHHEXCHR	8CAB	r	prüft auf Hexadezimalziffer
CHKCOM	AEFD	B	prüft, ob Komma folgt
CHKIN	FFC6	K	setzt Eingabekanal
CHKKLA	AEFA	B	prüft, ob Klammer auf folgt
CHKKLZ	AEF7	B	prüft, ob Klammer zu folgt
CHKOUT	FFC9	K	setzt Ausgabekanal
CHRGET	0073	B	holt nächstes Zeichen
CHRGOT	0079	B	holt letztes Zeichen
CHRGR	CB6E	v	Größe bei CHAR und TEXT
CIA1DDRA	DC02	p	CIA1 Datenrichtungsregister A
CIA1PRA	DC00	p	CIA1 Port A
CIA1PRB	DC01	p	CIA1 Port B
CIA2ICR	DD0D	p	CIA2 Interrupt-Control-Register
CIA2PRA	DD00	p	CIA2 Port A
CLOSE	FFC3	K	schliesst Datei
CLRCH	FFCC	K	deaktiviert alle Kanäle
CODEZAE	C5E4	v	Zähler für ZEIUMW-Routine
COPBYTE	C5E2	v	Sammelbyte für COPY-Befehl
COPZAEY	C5E6	v	Zähler (Y-Richtung) f. COPY
COSMALKRY	AA1F	r	Bildet COS(Winkel) * KRY
DELAY	C5F8	v	DELAY-Parameter (0-255)
DESTAB1	A71A	C	Tabelle f. DESIGN
DESTAB2	A712	C	Tabelle f. DESIGN
DESTAB3	A70A	C	Tabelle f. DESIGN
DETECTART	C515	v	DETECT-Parameter (0,1)
DETECTERG	C514	v	Kollisions-Byte
DIFFV	C5A1	w	Verhältnis von YDIFF zu XDIFF
DRAWTABX	CB59	V	Tabelle der Richtungsflags X
DRAWTABY	CB5D	V	Tabelle der Richtungsflags Y
DREHSINN	C57C	v	Drehsinn für ARC-Routine
DRGRZ	C503	v	Zähler f. Größe b. DRAW/ROT
DRINCX	CB65	v	Inkrement X-Richtung f. DRAW
DRINCY	CB69	v	Inkrement Y-Richtung f. DRAW
DRRICHT	CB61	v	Richtung für ROT/DRAW
DRSTATUS	CB50	v	Status von DRAW
DRSTRZ	CB4F	v	Zeiger in String von DRAW
DRX	CB6D	w	X-Koordinate für DRAW
DRY	C5A3	v	Y-Koordinate für DRAW
DRYANF	CB51	v	Anfangswert (Y) für DRAW
DSCBMOUT	9A5A	r	'ds-cbm' ausgeben
DSCBMOUT1	BDAA	r	'ds-cbm' ausgeben
DWERT	CBAE	v	Hilfsvariable
ENDE	BFBB		Ende von Simon's Basic

Symbolname	Hex.	T	Bedeutung
ENDSMB	822A	r	Simon's Befehl beenden
ERRLN	C51D	w	Zeilennummer, wo Fehler auftrat
ERRN	C5FC	v	Fehlercode von letztem Fehler
EX	C5AA	w	Endwert X-Koord. f. LINE (u.a.)
EY	C5A8	w	Endwert Y-Koord. f. LINE (u.a.)
FARBSET	9383	r	Farbe f. LOWCOL lokal ändern
FEHLMTAB	88BA	A	Tabelle der Fehlerm.-Adressen
FETCHANZ	CB6F	v	Anzahl der Zeichen für FETCH
FETCHLEN	CB72	v	Länge des Strings bei FETCH
FETCHT	B053	r	holt Zeichen von Tastatur
FETCHTAB	B15B	V	Tabelle der Sondercodes
FILLBER	A278	r	Bereich mit Code beschreiben
FLAG1	CA52	v	Hilfs-Flag
FLASHFL	C5C6	v	FLASH-Flag (aktiv = 10)
FLASHFLS	C504	V	Flag-Tabelle für FLASH-Farben
FLASHSP	C5C4	v	FLASH-Geschwindigkeit
FLASHZAE	C5C5	v	IRQ-Zähler für FLASH
FNAT	97B7	b	AT-Funktion
FNCHECK	90EF	b	CHECK-Funktion
FNDIV	8F8E	b	DIV-Funktion
FNDUP	8D69	b	DUP-Funktion
FNERR	9146	b	ERRor-Funktionen ERRLN und ERRN
FNERRLN	9158	b	ERRLN-Funktion
FNERRN	916D	b	ERRN-Funktion
FNEXOR	8D46	b	EXOR-Funktion
FNFRAC	8DFC	b	FRAC-Funktion
FNGRAPHICS	90CC	b	GRAPHICS-Funktion
FNINKEY	90A8	b	INKEY-Funktion
FNINSERT	8EB3	b	INSERT-Funktion
FNINST	9033	b	INST-Funktion
FNJOY	88D4	b	JOY-Funktion
FNLIN	8FEA	b	LIN-Funktion
FNMOD	8FBC	b	MOD-Funktion
FNPENX	893A	b	PENX-Funktion
FNPENY	8943	b	PENY-Funktion
FNPLACE	8E4F	b	PLACE-Funktion
FNPOT	891A	b	POT-Funktion
FNSOUND	90DA	b	SOUND-Funktion
FNTEST	8FFC	b	TEST-Funktion
FRQTABH	8C06	C	Frequenztabelle Low f. Musik
FRQTABL	8BF2	C	Frequenztabelle High f. Musik
GADRTABH	9446	C	Grafik-RAM-Adressen High
GADRTABL	942D	C	Grafik-RAM-Adressen Low
GBITCLR	9AB4	r	löscht Bit im Grafik-RAM
GBITSET	9AAB	r	setzt Bit im Grafik-RAM
GBITTABL	9340	C	Tabelle zum Löschen von Bits
GBITTABS	9330	C	Tabelle zum Setzen von Bits

Symbolname	Hex.	T	Bedeutung
GET2NYB	99CC	r	holt zwei Werte (0-15)
GETBIN	8CC3	r	holt Binärzahl
GETBYTC	8C75	r	prüft auf Komma und holt Byte
GETHEX	8CD7	r	holt Hexadezimal-Wert
GETIN	FFE4	K	holt ein Zeichen aus Tast.puff.
GETPKTF	937D	r	holt Punktfarbe (nach $F7)
GETRCWD	AC8D	r	holt 4 Parameter
GETXYXYF	9359	r	holt 5 Parameter
GFLAG	C5AE	v	Flag f. Block-Zeichnen
GMEMFLAG	C5B3	v	Flag, für Grafik-Modus
HEX2W	8D28	r	holt Wert von 2-stell. Hexzahl
HEXCON	8CE0	r	wandelt Hexzahl nach Integer
HEXW	8D3C	r	bildet Wert von Hex-Ziffer
HILFFL1	C52B	v	Hilf-Flag f. PAINT
HILFFLAG	C52C	v	Hilf-Flag f. PAINT
IDIV	8F45	r	ganzzahlige Division
IDIVDEND	CBC0	w	Dividend und Ergebnis f. IDIV
IDIVREST	CBC2	w	Rest von IDIV
IDIVSOR	CBBB	w	Divisor für IDIV
IFFLAG	C5CB	v	Ergebnis der IF-Bedingung
INC$20	A039	r	$20,$21 um eins erhöhen
INC$20	9CDC	r	$20,$21 um eins erhöhen
INCA6A7	9AE9	r	$A6,$A7 um eins erhöhen
INCA8A9	9AF0	r	$A8,$A9 um eins erhöhen
INCBASBZ	83DB	r	Basic-Befehlszeiger erhöhen
INKEY	C5D6	v	Nummer von Funktionstaste
JOYWERT	CB70	v	Wert von JOY-Abfrage (u.a.)
KERERRB	E0F9	B	Behandlung von Kernal-Fehlern
KERROMAUS	9406	r	Kernal-ROM ausblenden (HIRAM=0)
KERROMEIN	940E	r	Kernal-ROM einsch. (HIRAM=1)
KEYFLAG	C646	v	Flag, ob KEY gesperrt (aktiv=0)
KEYON	C5EC	v	gedrückte Taste bei ONKEY1
KEYTAB	C64D	V	Tabelle der Strings f. KEY
KMX	CB4B	w	Kreismittelpunkt X-Koord.
KMXMINUS	AAB0	r	Wert von Kreismp.X abziehen
KMXPLUS	AA97	r	Wert zum Kreismp.X addieren
KMY	CB49	w	Kreismittelpunkt Y-Koord.
KMYMINUS	AAC8	r	Wert von Kreismp.Y abziehen
KMYPLUS	AAA7	r	Wert zum Kreismp.Y addieren
KREIS	A914	r	Ellipse bzw. Polygon zeichnen
KREIS1	A94F	r	Fortsetzung von KREIS
KRX	CB48	v	Ellipsen-Radius X-Komponente
KRY	CBC5	v	Ellipsen-Radius Y-Komponente
LASTKEY	C642	v	Letzte gedrückte Taste
LCFARB12	CB1B	v	Farben 1 und 2 von LOW COL
LCFARB3	CB1C	v	Farbe 3 für LOW COL
LCODE	C5D5	v	Hilfszähler für LIST-Routine

Symbolname	Hex.	T	Bedeutung
LINZEI	A35E	r	Linie ziehen
LOCALTAB	CA53	V	Tabelle der als LOCAL def. Var.
LOWCOLFLAG	CB1D	v	LOWCOL-Flag (aktiv = 10)
LPX	C500	v	Wert von Light-Pen (X-Anteil)
LPY	C501	v	Wert von Light-Pen (Y-Anteil)
LXMERK	03FB	v	Zwischenspeicher f. X bei LIST
LYMERK	03FA	v	Zwischensp. f. Y bei LIST-Rout.
MBITTABL	9780	C	Tabelle zum Löschen von Bits
MBITTABS	9778	C	Tabelle zum Setzen von Bits
MELDUNG	8208	T	Einschalt-Meldung
MERKEPKT	B9D7	r	Punkt auf PAINT-Stapel legen
MERKZEI	B49B	r	Zeichen im Puffer merken (USE)
MFREQ	C599	w	Frequenzwert bei Musik
MOBBEW	C5B4	v	Flag, ob MOB bewegt werden soll
MOBBEWEG	A3AC	r	MOB bewegen bzw. Linie ziehen
MOBEXL	8C5B	r	MOB in X-Richt. verkleinern
MOBEXS	8C41	r	MOB in X-Richt. vergrößern
MOBEYL	8C68	r	MOB in Y-Richt. verkleinern
MOBEYS	8C4E	r	MOB in Y-Richt. vergrößern
MOBNR	C5BC	v	Nummer des Sprites
MOBNR2	C5BB	v	Doppelte Spritenummer
MOBPOSS	9625	r	Setze neue MOB-Position
MOBSPEED	C5B5	v	Geschwindigkeit des MOB's
MODULTEXT	8004	T	Text zur Modul-Erkennung
MOVEBEF	AC5D	b	Alle Move-und Scroll-Befehle
MOVEDR	AEC6	r	Move nach unten oder rechts
MOVETAB	CB1E	V	Platz f. alte Zeile bzw. Spalte
MOVEUL	AD8D	r	Move nach oben oder links
MOVEZ	ACB5	v	Zähler für Move-Befehle
MREGADR	CB8C	w	Basis-Adresse im SID f. Stimme
MULTIJN	C5B0	v	Multi-Flag (aktiv = $2C)
MUSICADR	CB9B	w	Adresse des Strings nach MUSIC
MUSICDAU	CB96	v	Ton-Dauer-Parameter von MUSIC
MUSICLEN	CB9E	v	Länge des Strings nach MUSIC
MUSICZ1	CB9D	v	Zähler für Musik (SIRQPALY)
MUSICZ2	CB86	v	Zähler für Musik
MUSICZ3	CB88	v	Zähler für Musik
MUSICZ4	CB8A	v	Zähler für Musik
MUSICZ5	CB9F	v	Zähler für Musik
NEUSTR	B47D	B	Neuen String einrichten
NOTENTAB	8BDE	T	Tab. mit den zul. Notenzeichen
NOTESUCH	8BA1	r	suche Note in NOTENTAB
NRM	A837	r	normiert Video-Register
ONERRORFLAG	C5FB	v	ON ERROR-Flag (aktiv = 10)
ONERRZEIG	C5F9	w	Zeiger auf Error-Routine
ONKEY1	9DBA	r	behandelt ONKEY bei IRQ
ONKEYFLAG	C5EA	v	ON KEY-Flag (aktiv = 10)

Symbolname	Hex.	T	Bedeutung
ONKEYZEIG	C5E8	w	Zeiger auf ONKEY-Routine
OPEN	FFC0	K	öffnet Datei
OPTFLAG	C5DC	v	OPTION-Flag (aktiv = 10)
OUT=	A040	r	Gleichheitszeichen ausgeben
OUTASC	A02B	r	Text ab $0100 ausgeben
PAGEFLAG	C58C	v	PAGE-Flag (aktiv = 10)
PAGEWERT	C58D	v	Zeilenzahl für PAGE
PAKTY	CA51	v	akt. Y-Koord. bei PAINT
PENABFR	894E	r	Light-Pen-Abfrage
PENDFLAG	C5DD	v	Flag, ob Programm zu Ende
PKBER	AA71	r	absolute Koord. berechnen
PLAYEND	8ADE	r	Musik als beendet kennzeichnen
PLAYFLAG	CB91	v	spielt gerade Musik ? (ja=10)
PLPOS	CBB1	v	Position bei PLACE
PORTSPEI	C596	v	Speicher für alten Port-Wert
POS	C51F	v	Positionszäh. bei String-Bef.
POS1	C520	v	Positionszäh. bei String-Bef.
PRESTOR	813B	r	stellt alten Wert im Port her
PROCNFFL	C597	v	Flag, ob 'proc not f.' ausgeben
PSPEIBE	812C	r	Port-Wert sichern und Basic ein
PSPEINMI	C58B	v	Port-Wert bei NMI
PTEST1	A261	r	Testet Punkt im Hires-Modus
PUNKT	9297	r	zeichnet einen Punkt
PUSHSTR	B4CA	B	bringt String-Deskr. auf Stapel
PXHTAB	C751	V	Stapel der X-K. (high) f. PAINT
PXLTAB	C850	V	Stapel der X-K. (low) f. PAINT
PYALT	CA50	v	alter Y-Wert bei PAINT
PYMERK	CA4F	v	Merker für Y bei Paint
PYTAB	C94F	V	Stapel der Y-K. für PAINT
PZEIG	CA4E	v	Zeiger in PAINT-Stapel
PZEILB	01FB	V	Bereich für Programmzeile
RICHTZ	CB7C	v	Richtungsflag u. Zäh. b. Moveb.
ROTGR	C502	v	Größe bei ROT/DRAW
RVSBLANK	B170	r	ein RVS-Leerzeichen ausgeben
SASCFLP	8079	r	Text in Fliesskomma wandeln
SBASBEF	917A	r	Befehl dekodieren und ausführ.
SBRK	8314	b	BRK-Routine von Simon's Basic
SBZPLY	80FD	r	Basic-Zeiger um Y erhöhen
SCHKCOM	8031	r	prüft, ob Komma folgt
SCHKCOM1	8067	r	prüft, ob Komma folgt
SCHKKLZ	807F	r	prüft, ob Klammer zu folgt
SCHKZEI	805E	r	prüft, ob Code im Akku folgt
SCHRART	CB4D	v	Schriftart für TEXT
SCLRCH	80D0	r	schliesst alle Kanäle
SCOS	8028	r	ruft Basic-SIN-Funktion auf
SERROUT	888C	r	gibt Simon's-Fehlermeldung aus
SETFNPAR	FFBD	K	setzt Filenamenparameter

Symbolname	Hex.	T	Bedeutung
SETFPAR	FFBA	K	setzt Fileparameter
SFACADR	8016	r	wandelt FAC in Adressformat
SFACARG	8091	r	überträgt FAC nach ARG
SFACASC	80B5	r	wandelt FAC in Text um
SFDIV	80AC	r	ruft Basic-Fliessk.Division auf
SFLDFACK	80A3	r	lädt FAC mit Konstante
SFMULT	809A	r	ruft Basic-Fliessk.-Mult. auf
SFRESTR	8043	r	holt Stringparameter (n.FRMEVL)
SFRMEVL	803A	r	wertet beliebigen Ausdruck aus
SFRMNUM	800D	r	wertet numerischen Ausdruck aus
SGETADR	9427	r	holt Adressparameter
SGETADR2	9415	r	holt Adressparameter
SGETARI	8C8D	r	wertet arithmet. Element aus
SGETARI1	807C	r	ruft Basic-GETARI-Routine auf
SGETBYT1	8106	r	holt Byte-Parameter
SGETBYTC	81FC	r	prüft auf Komma und holt ByteP
SGETBYTN	8202	r	überliest Code und holt Byte-P
SGETSTR	86B7	r	holt Stringparameter
SGETSTR1	8C7B	r	holt Stringparameter
SGETSTRN	86B4	r	überl. Code u. holt Stringpar.
SGETVAR	80BE	r	holt Variablenparameter
SGOTO1	8055	r	zum Basic-GOTO-Befehl
SGOTO11	810F	r	zum Basic-GOTO-Befehl
SHCOCTFL	028D	s	Flag f. Shift/C=/CTRL
SHWERTTAB	9892	C	Wert von Sh./C= für KEY
SIDTAB	9A54	C	Tab. d. Basisadr. d. SID / St.
SINMALKRX	AA01	r	bildet SIN(Winkel) * KRX
SINPUT1	80EB	r	zum Basic-INPUT-Befehl
SINTOUT	804C	r	gibt Integer-Zahl aus
SIRQ	9694	b	Simon's IRQ-Routine (BFLASH)
SIRQ2	97D2	r	Fortsetzung (FLASH)
SIRQ3	9850	r	Fortsetzung (INKEY)
SIRQ4	9896	r	Fortsetzung (KEY)
SIRQ5	98FF	r	Fortsetzung (PLAY)
SIRQPLAY	8A61	r	IRQ-Unterroutine für PLAY
SLIST	8240	b	Simon's List-Routine (Teil 1)
SLIST1	824A	r	zur Basic-List-Routine
SLIST2	8253	r	zur Basic-List-Schleife
SLIST3	BECB	r	Simon's List-Routine (Teil 2)
SLOOP	825C	b	Eingabeschleife (Simon's Vers.)
SNEXTTR	80E2	r	sucht nächstes Trennzeichen
SNMI	82FA	b	Simon's NMI-Routine
SNMIVEC	8002	S	NMI-Vektor im Simon's Modul
SOPEN	8123	r	ruft Basic-Befehl OPEN auf
SPALTEANF	CB77	v	Anfangsspalte für Movebefehle
SPALTENANZ	CB78	v	Soaltenzahl für Movebefehle
SPEXEC	C62C	v	Stackpointer für EXEC

Symbolname	Hex.	T	Bedeutung
SPLOOP	C641	v	Stackpointer für LOOP
SPREPEAT	C617	v	Stackpointer für REPEAT
SPRTAB	86C4	A	Sprungtabelle für Befehle
SPUFOCR	80F4	r	Puffer mit 0 abschl. und Return
SREM	80D9	r	zum Basic-Befehl REM
SRTAB	CBA0	V	Tabelle für Sustain/Release
SSGETADR	9415	r	holt Adressparameter
SSIN	801F	r	ruft Basic-Funktion SIN auf
SSMNI	8118	b	NMI-Einsprung, wenn KERNAL aus
SSTART	800A	b	Kaltstart-Adresse von Sim.B.
SSTRPZ	8070	r	berechnet Zeilenstartadresse
STACKEXEC	C618	W	Stack für EXEC
STACKLOOP	C62D	W	Stack für LOOP
STACKREPEAT	C603	W	Stack für REPEAT
STARTVEC	8000	S	Modul-Reset-Vektor
STESTDIRM	80C7	r	pruft auf Programm-Modus
STIMMEIN	8C1A	r	schletet Stimme ein
STIMMENR	CBA8	v	Nummer der akt. Stimme
STR1	CBAB	v	Hilfsadresse bei Stringbefehlen
STRLEN1	CBB7	v	Stringlge. d. 1.Str. b. Strbef.
STRLEN2	CBB4	v	Stringlge. d. 2.Str. b. Strbef.
STRZ1	C518	v	Hilfszelle für String-Befehle
STRZ2	C519	v	Hilfszelle für String-Befehle
SUCHCODE	9C39	r	sucht Code im Programm
SUCHCODEN	9C60	r	überliest Code und sucht dann
SWARM	9E9C	b	Simon's Warmstart
SXFLP	8088	r	wandelt nach Fliessk. (X=Exp.)
SXYFLP	8CCD	r	wandelt Zahl in (X/Y) nach Flk.
TAB?	86AA	V	fragwürdige Tabelle
TABBEF	83E2	T	Tabelle der Befehlswörter
TABJOY	8912	C	Tabelle der Joystick-Werte
TABROT	A11E	C	Tab. für Richtungen bei DRAW
TABSAUBER	B96E	r	'säubert' Stapel für PAINT
TABSAUBER1	B975	r	Fortsetzung von TABSAUBER
TASTPUFF	0277	s	Tastaturpuffer
TESTFREI	B9BA	r	prüft,ob Punkt gesetzt (PAINT)
TEXTCHR	BE0A	T	Text "CHR$(13)"
TEXTKEY	BE06	T	Text "KEY"
TRACE1	896A	r	behandelt TRACE
TRACEFLAG	C601	v	TRACE-Flag (10 = aktiv)
TRACESH1	89E9	r	Fortsetzung von TRACESHOW
TRACESHOW	89E2	r	beschreibt TRACE-Fenster
TRACETAB	C531	V	Tabelle der letzten Zeilennrn.
TRACETAB+36	C555	V	letzte Zeilennummer
TRSHIFT	8998	r	schiebt Znrn in TRACETAB weiter
UEBER4	A6FE	r	überliest 4 Byte des Programms
UEBERDREH	C52F	v	Flag, wenn 360 Grad überschr.

Symbolname	Hex.	T	Bedeutung
USELEN	020D	v	Länge des Strings bei USE
USEPUFF	0212	v	Resultat-String bei USE
VGLA	C523	w	Vergleichswert 1
VGLAE	A616	r	Vergleichsroutine
VGLE	C51A	w	Vergleichswert 2
VICADS	D018	p	VIC Adressregister
VICHIFAR	D021	p	VIC Hintergrundfarbe 0
VICIRQF	D019	p	VIC Interrupt-Flag-Register
VICIRQM	D01A	p	VIC Interrupt-Maske
VICLPX	D013	p	VIC Light-Pen-X Register
VICLPY	D014	p	VIC Light-Pen-Y Register
VICMEA	D015	p	VIC MOB's ein/aus
VICMEX	D01D	p	VIC MOB's vergr. X-Richtung
VICMEY	D017	p	VIC MOB's vergr. Y-Richtung
VICMFA	D027	p	VIC MOB-Farbe (MOB 0)
VICMMC	D01C	p	VIC MOB-Art (Multi ?)
VICMMC1	D025	p	VIC MOB-Farbe 1
VICMMC2	D026	p	VIC MOB-Farbe 2
VICMPR	D01B	p	VIC MOB-Priorität
VICMX	D000	p	VIC MOB-Koordinaten X (Bit 0-7
VICMX8	D010	p	VIC MOB-Koordinaten X (Bit 8)
VICMY	D001	p	VIC MOB-Koordinaten Y
VICRAFAR	D020	p	VIC Rahmenfarbe
VICST1	D011	p	VIC Steuerregister 1
VICST2	D016	p	VIC Steuerregister 2
VIDRAMHI	0288	s	High-Byte Beginn Video-RAM
WARM	E386	B	normaler Basic-Warmstart
WAVETAB	C64A	V	Tabelle der Wellenformen
WEISSECK	8A3A	r	Weisses Fenster f. TRACE malen
WINKSW	CB4D	w	Winkel-Schrittweite f. KREIS
X123	9A6F	r	prüft, ob X zwischen 1 und 3
XDIFF	C5A6	w	Differenz der X-Koordinaten
XMAXHIGH	C52E	v	Max. X-Koord. (High-Byte)
XMAXLOW	C52D	v	Max. X-Koord. (Low-Byte)
XNYB	9A6A	r	prüft, ob X kleiner 16
YDIFF	C59F	w	Differenz der Y-Koordinaten
YNEXT	A50C	r	peilt nächsten Y-Wert an
ZAEHLER	CB7A	v	Hilfszähler
ZAEHLM1	CB7E	v	Hilfszähler
ZAEHLM2	CB80	v	Hilfszähler
ZAEIRQ	C516	v	Zähler für 1/60 sec.
ZAESEC	C517	v	Zähler für Sekunden (PAUSE)
ZEILEANF	CB76	v	Anfangszeile für Movebefehle
ZEILENANZ	CB79	v	Zeilenanzahl für Movebefehle
ZEIUMW	8234	b	Umwandlung einer Prog.-Zeile
ZEIUMW1	AB79	b	Fortsetzung von ZEIUMW
ZN	C586	w	Zeilennummer

Symbolname	Hex.	T	Bedeutung
ZWISP2	CB54	v	Hilfszelle
ZWISP3	CB84	v	Hilfszelle
ZWISP4	CB71	v	Hilfszelle
ZWSPEI1	C57D	v	Hilfszelle

0073	CHRGET	80D9	SREM
0079	CHRGOT	80E2	SNEXTTR
01FB	PZEILB	80EB	SINPUT1
01FE	AUTOZI	80F4	SPUFOCR
01FF	AUTOZI+1	80FD	SBZPLY
020D	USELEN	8106	SGETBYT1
0212	USEPUFF	810F	SGOTO11
0277	TASTPUFF	8118	SSMNI
0286	AKTFARB	8123	SOPEN
0288	VIDRAMHI	812C	PSPEIBE
028D	SHCOCTFL	813B	PRESTOR
03FA	LYMERK	8147	BEFCOLD
03FB	LXMERK	81FC	SGETBYTC
8000	STARTVEC	8202	SGETBYTN
8002	SNMIVEC	8208	MELDUNG
8004	MODULTEXT	822A	ENDSMB
800A	SSTART	8234	ZEIUMW
800D	SFRMNUM	8240	SLIST
8016	SFACADR	824A	SLIST1
801F	SSIN	8253	SLIST2
8028	SCOS	825C	SLOOP
8031	SCHKCOM	82EC	BASROMAU
803A	SFRMEVL	82F3	BASROMEI
8043	SFRESTR	82FA	SNMI
804C	SINTOUT	8314	SBRK
8055	SGOTO1	8344	BEFMERGE
805E	SCHKZEI	837F	BEFRENUM
8067	SCHKCOM1	83DB	INCBASBZ
8070	SSTRPZ	83E2	TABBEF
8079	SASCFLP	86AA	TAB?
807C	SGETARI1	86B4	SGETSTRN
807F	SCHKKLZ	86B7	SGETSTR
8088	SXFLP	86C4	SPRTAB
8091	SFACARG	86C5	SPRTAB+1
809A	SFMULT	888A	BADMODE
80A3	SFLDFACK	888C	SERROUT
80AC	SFDIV	88BA	FEHLMTAB
80B5	SFACASC	88BB	FEHLMTAB+1
80BE	SGETVAR	88D4	FNJOY
80C7	STESTDIRM	8912	TABJOY
80D0	SCLRCH	891A	FNPOT

893A	FNPENX	9297	PUNKT
8943	FNPENY	9330	GBITTABS
894E	PENABFR	9340	GBITTABL
896A	TRACE1	9350	BEFLINE
8998	TRSHIFT	9359	GETXYXYF
89E2	TRACESHOW	937D	GETPKTF
89E9	TRACESH1	9383	FARBSET
8A3A	WEISSECK	93DC	BEFLOWCOL
8A61	SIRQPLAY	9406	KERROMAUS
8ADE	PLAYEND	940E	KERROMEIN
8BA1	NOTESUCH	9415	SSGETADR
8BDE	NOTENTAB	9415	SGETADR2
8BF2	FRQTABL	9427	SGETADR
8C06	FRQTABH	942D	GADRTABL
8C1A	STIMMEIN	9446	GADRTABH
8C41	MOBEXS	945F	A4MAL40
8C4E	MOBEYS	9489	BEFCIRCLE
8C5B	MOBEXL	94CE	20PL40
8C68	MOBEYL	94DC	23PL40
8C75	GETBYTC	94EA	BEFMULTI
8C7B	SGETSTR1	9537	BEFCOLOUR
8C8D	SGETARI	9546	BEFDIR
8CAB	CHHEXCHR	95C8	BEFMMOB
8CC3	GETBIN	9625	MOBPOSS
8CCD	SXYFLP	9669	BEFBFLASH
8CD7	GETHEX	9694	SIRQ
8CE0	HEXCON	96DA	BEFMOBSET
8D28	HEX2W	9759	BEFMUSIC
8D3C	HEXW	9778	MBITTABS
8D46	FNEXOR	9780	MBITTABL
8D69	FNDUP	9788	BITWTAB
8DFC	FNFRAC	9790	BEFFLASH
8E4F	FNPLACE	97B7	FNAT
8EB3	FNINSERT	97D2	SIRQ2
8F45	IDIV	9836	BEFPAGE
8F8E	FNDIV	9850	SIRQ3
8FBC	FNMOD	9892	SHWERTTAB
8FEA	FNLIN	9896	SIRQ4
8FFC	FNTEST	98FF	SIRQ5
9033	FNINST	9918	BEFPLAY
90A8	FNINKEY	9947	BEFCENTRE
90CC	FNGRAPHICS	9979	BEFENVELOPE
90DA	FNSOUND	99CC	GET2NYB
90EF	FNCHECK	99E8	BEFCGOTO
9146	FNERR	99F7	BEFWAVE
9158	FNERRLN	9A09	BINCON
916D	FNERRN	9A54	SIDTAB
9174	BEFO	9A5A	DSCBMOUT
917A	SBASBEF	9A6A	XNYB
91FF	BEFHIRES	9A6F	X123
9267	BEFPLOT	9AAB	GBITSET

9AB4	GBITCLR
9AE9	INCA6A7
9AF0	INCA8A9
9AF7	BEFREPEAT
9B15	BEFUNTIL
9B48	BEFRETRACE
9B5A	BEFTRACE
9B63	BEFOPTION
9B6C	BEFIF
9BD6	BEFAUTO
9C05	BEFRESET
9C2A	BEFCALL
9C39	SUCHCODE
9C60	SUCHCODEN
9CDC	INC$20
9CE3	BEFEXEC
9D19	BEFENDPROC
9D37	BEFEXIT
9D6F	BEFENDLOOP
9D89	BEFONKEY
9DB2	BEFDISABLE
9DBA	ONKEY1
9E0F	BEFRESUME
9E24	BEFLOOP
9E42	BEFDELAY
9E4B	BEFSECURE
9E6E	BEFONERR
9E94	BEFNOERR
9E9C	SWARM
9ED1	BEFOUT
9EDA	BEFOLD
9F1F	BEFRCOMP
9F30	BEFPROC
9F3F	BEFDUMP
A002	BNMIVEC
A02B	OUTASC
A039	INC$20
A040	OUT=
A045	CHAREN1
A04C	CHAREN0
A057	BEFDRAW
A11E	TABROT
A15E	BEFROT
A186	BEFCHAR
A1B4	CHARZEI
A261	PTEST1
A270	BEFHICOL
A278	FILLBER
A297	BEFFILL
A2D0	BEFFCHR
A2E2	BEFFCOL
A2F4	BEFLOCAL
A33C	BEFGLOBAL
A35E	LINZEI
A3AC	MOBBEWEG
A50C	YNEXT
A594	BEFFIND
A616	VGLAE
A62E	BEFDESIGN
A6FE	UEBER4
A70A	DESTAB3
A712	DESTAB2
A71A	DESTAB1
A768	BEFRLOCM
A7A6	BEFCMOB
A7B5	BEFBCKGNDS
A7E2	BEFPAUSE
A831	BEFNRM
A837	NRM
A856	BEFMOBOFF
A865	BEFOFF
A876	BEFARC
A88D	ARC1
A914	KREIS
A94F	KREIS1
AA01	SINMALKRX
AA1F	COSMALKRY
AA3D	A8A9BOGM
AA71	PKBER
AA97	KMXPLUS
AAA7	KMYPLUS
AAB0	KMXMINUS
AAC8	KMYMINUS
AAD7	A8A9DIV90
AB10	BEFANGL
AB79	ZEIUMW1
AC5D	MOVEBEF
AC8D	GETRCWD
ACB5	MOVEZ
AD8D	MOVEUL
AEC6	MOVEDR
AEF7	CHKKLZ
AEFA	CHKKLA
AEFD	CHKCOM
B004	BEFFETCH
B053	FETCHT
B15B	FETCHTAB
B170	RVSBLANK
B18F	BEFSCRSV
B1DC	BEFSCRLD
B229	BEFTEXT
B30D	BEFCSET

B33F	BEFVOL	C57C	DREHSINN
B35D	BEFDISK	C57D	ZWSPEI1
B393	BEFUSE	C586	ZN
B47D	NEUSTR	C587	ZN+1
B49B	MERKZEI	C58B	PSPEINMI
B4A0	BEFHRDCPY	C58C	PAGEFLAG
B4CA	PUSHSTR	C58D	PAGEWERT
B590	BEFKEY	C596	PORTSPEI
B5E8	BEFPAINT	C597	AX;PROCNFFL
B96E	TABSAUBER	C598	AX+1
B975	TABSAUBER1	C599	MFREQ
B9BA	TESTFREI	C59A	MFREQ+1
B9D7	MERKEPKT	C59F	YDIFF
B9F1	BEFCOPY	C5A0	YDIFF+1
BAF7	BEFREC	C5A1	DIFFV
BC6E	BEFBLOCK	C5A2	DIFFV+1
BC77	BLOCKZEI	C5A3	DRY
BDAA	DSCBMOUT1	C5A6	XDIFF
BDAD	BEFMEM	C5A7	XDIFF+1
BDF7	BEFDETECT	C5A8	EY
BE06	TEXTKEY	C5A9	EY+1
BE0A	TEXTCHR	C5AA	EX
BE13	BEFDISPLAY	C5AB	EX+1
BECB	SLIST3	C5AC	AY
BFBB	ENDE	C5AD	AY+1
C500	LPX	C5AE	GFLAG
C501	LPY	C5B0	MULTIJN
C502	ROTGR	C5B1	AUTOINC
C503	DRGRZ	C5B2	AUTOFLAG
C504	FLASHFLS	C5B3	GMEMFLAG
C514	DETECTERG	C5B4	MOBBEW
C515	DETECTART	C5B5	MOBSPEED
C516	ZAEIRQ	C5BB	MOBNR2
C517	ZAESEC	C5BC	MOBNR
C518	STRZ1	C5C4	FLASHSP
C519	STRZ2	C5C5	FLASHZAE
C51A	VGLE	C5C6	FLASHFL
C51B	VGLE+1	C5CB	IFFLAG
C51D	ERRLN	C5D5	LCODE
C51E	ERRLN+1	C5D6	INKEY
C51F	POS	C5D7	BFLASHSP
C520	POS1	C5D8	BFLZAE
C523	VGLA	C5D9	BFLASHF1
C524	VGLA+1	C5DA	BFLASHF2
C52B	HILFFL1	C5DB	BFLASHJN
C52C	HILFFLAG	C5DC	OPTFLAG
C52D	XMAXLOW	C5DD	PENDFLAG
C52E	XMAXHIGH	C5E2	COPBYTE
C52F	UEBERDREH	C5E4	CODEZAE
C531	TRACETAB	C5E6	COPZAEY
C555	TRACETAB+36	C5E8	ONKEYZEIG

C5E9	ONKEYZEIG+1	CB65	DRINCX
C5EA	ONKEYFLAG	CB69	DRINCY
C5EC	KEYON	CB6D	DRX
C5F0	BASBZK	CB6E	DRX+1;CHGR
C5F1	BASBZK+1	CB6F	FETCHANZ
C5F8	DELAY	CB70	JOYWERT
C5F9	ONERRZEIG	CB71	ZWISP4
C5FA	ONERRZEIG+1	CB72	FETCHLEN
C5FB	ONERRORFLAG	CB76	ZEILEANF
C5FC	ERRN	CB77	SPALTEANF
C601	TRACEFLAG	CB78	SPALTENANZ
C603	STACKREPEAT	CB79	ZEILENANZ
C617	SPREPEAT	CB7A	ZAEHLER
C618	STACKEXEC	CB7C	RICHTZ
C62C	SPEXEC	CB7E	ZAEHLM1
C62D	STACKLOOP	CB80	ZAEHLM2
C641	SPLOOP	CB82	BWFLAG
C642	LASTKEY	CB84	ZWISP3
C646	KEYFLAG	CB86	MUSICZ2
C64A	WAVETAB	CB88	MUSICZ3
C64D	KEYTAB	CB8A	MUSICZ4
C751	PXHTAB	CB8C	MREGADR
C752	PXHTAB+1	CB8D	MREGADR+1
C850	PXLTAB	CB91	PLAYFLAG
C851	PXLTAB+1	CB96	MUSICDAU
C94F	PYTAB	CB9B	MUSICADR
C950	PYTAB+1	CB9C	MUSICADR+1
CA4E	PZEIG	CB9D	MUSICZ1
CA4F	PYMERK	CB9E	MUSICLEN
CA50	PYALT	CB9F	MUSICZ5
CA51	PAKTY	CBA0	SRTAB
CA52	FLAG1	CBA4	ADTAB
CA53	LOCALTAB	CBA8	STIMMENR
CB1B	LCFARB12	CBAB	STR1
CB1C	LCFARB3	CBAC	STR1+1
CB1D	LOWCOLFLAG	CBAE	DWERT
CB1E	MOVETAB	CBB1	PLPOS
CB48	KRX	CBB4	STRLEN2
CB49	KMY	CBB7	STRLEN1
CB4A	KMY+1	CBBB	IDIVSOR
CB4B	KMX	CBBC	IDIVSOR+1
CB4C	KMX+1	CBC0	IDIVDEND
CB4D	WINKSW;SCHRART	CBC1	IDIVDEND+1
CB4E	WINKSW+1	CBC2	IDIVREST
CB4F	DRSTRZ	CBC3	IDIVREST+1
CB50	DRSTATUS	CBC5	KRY
CB51	DRYANF	D000	VICMX
CB54	ZWISP2	D001	VICMY
CB59	DRAWTABX	D010	VICMX8
CB5D	DRAWTABY	D011	VICST1
CB61	DRRICHT	D013	VICLPX

D014	VICLPY	DC02	CIA1DDRA
D015	VICMEA	DD00	CIA2PRA
D016	VICST2	DD0D	CIA2ICR
D017	VICMEY	E0F9	KERERRB
D018	VICADS	E386	WARM
D019	VICIRQF	EA31	BIRQ
D01A	VICIRQM	FFBA	SETFPAR
D01B	VICMPR	FFBD	SETFNPAR
D01C	VICMMC	FFC0	OPEN
D01D	VICMEX	FFC3	CLOSE
D020	VICRAFAR	FFC6	CHKIN
D021	VICHIFAR	FFC9	CHKOUT
D025	VICMMC1	FFCC	CLRCH
D026	VICMMC2	FFCF	BASIN
D027	VICMFA	FFD2	BSOUT
DC00	CIA1PRA	FFE4	GETIN
DC01	CIA1PRB		

6.4 Die wichtigsten Zero-Page-Adressen

Hier werden nur die wichtigsten Belegungen der Zero-Page-Adressen aufgeführt, und auch nur die, die sich in der Anwendung wesentlich vom normalen Basic unterscheiden.

Allgemeine Hilfszellen:

Adresse	Bedeutung
\$09,\$0A	Hilfszeiger
\$20 bis \$24	diverse Zeiger und Zwischenspeicher
\$63,\$62	Low- und High-Byte zur Umwandlung mit SXFLP
\$69	Stringlänge nach SGETSTR
\$A4 bis \$AB	diverse Zeiger und Zwischenspeicher

Spezielle Hilfszellen / Parameter:

Adresse	Bedeutung	für Routine
\$09,\$0A	X-Koordinate	PUNKT
\$20,\$21	Adresse, wo Code gefunden	SUCHCODE
\$21	Flag für mehrere Punkte berechn.	KREIS
\$22	zu suchender Code	SUCHCODE
\$57 bis \$5E	acht (2*4) Umgebungsflags	PAINT
\$5F,\$60	akt. X-Koordinate	PAINT,COPY
\$5F	Stringlänge	DRAW
\$61	akt. Y-Koordinate	PAINT,COPY
\$62	Fertig-Flag	PAINT
\$65	Stapelzeiger	PAINT

$66	Füllcode	FILL,FCHR,FCOL
$66	Sollfarbe	PAINT
$6A	Zähler	CENTRE (u.a.)
$6A	Farbe des letzten Nachbarn	PAINT
$90	Status (Erg. von TEST, bzw. 8)	PUNKT
$A4	Y-Koordinate	PUNKT
$A6	Flag, ob Farb-RAM gemeint ist	FILL,FCHR,FCOL
$A8,$A9	Adresse im Grafik-RAM	PUNKT
$A8,$A9	Startwinkel	KREIS
$A8	Sammelbyte	DESIGN
$AA	Wertepaar (2 Werte bis 15)	GET2NYB
$AC,$AD	Endwinkel	KREIS
$AC	akt. Farbkombination	HIRES
$F7	Punktfarbe	Grafik-Befehle
$FC,$FD	Adresse Music-String	SIRQPLAY
$FE,$FF	Basis-Adresse f. Stimmregister	SIRQPLAY

Anhang

Anhang 1 : RAKETE 3 - Beispiel aus Band 3

```
10 IFL=0THENL=1:LOAD"BASERW3.OBJ",8,1
20 IFL=1THENL=2:LOAD"GRA3.OBJ",8,1
30 IFL=2THENL=3:LOAD"PAUSE.OBJ",8,1
40 IFL=3THENL=4:LOAD"SMOVE.OBJ",8,1
50 SYS51200
60 L=0
100 REM ****************************************
101 REM *        R A K E T E / S P R I T E S    *
102 REM ****************************************
110 POKE53280,6
120 POKE53281,6
130 V=53248
140 DIMH$(7)
150 POKEV+37,14
160 POKEV+38,0
997 REM ****************************************
998 REM *   DATAS FUER SPRITE 0                 *
999 REM ****************************************
1000 DATA0,20,0,0,20,0,0,20,0
1010 DATA0,150,0,0,150,0,0,150,0
1020 DATA0,150,0,0,150,0,48,150,12
1030 DATA48,150,12,50,150,140,58,150,172
1040 DATA58,150,172,58,150,172,186,150,174
1050 DATA186,150,174,186,150,174,186,150,174
1060 DATA1,20,64,1,20,64,1,20,64
1097 REM ****************************************
1098 REM *   DATAS FUER SPRITE 1                 *
1099 REM ****************************************
1100 DATA12,0,48,12,0,48,12,0,48
1110 DATA12,0,48,12,0,48,12,0,48
1120 DATA12,0,48,12,0,48,12,0,48
1130 DATA12,0,48,12,0,48,12,0,48
1140 DATA12,0,48,12,0,48,12,0,48
1150 DATA12,0,48,12,0,48,12,0,48
1160 DATA12,0,48,12,0,48,12,0,48
1197 REM ****************************************
1198 REM *   DATAS FUER SPRITE 2                 *
1199 REM ****************************************
1200 DATA192,0,3,192,0,3,192,0,3
1210 DATA192,0,3,192,0,3,192,0,3
1220 DATA192,0,3,192,24,3,192,60,3
1230 DATA192,102,3,255,231,255,192,102,3
1240 DATA192,60,3,192,24,3,192,0,3
1250 DATA192,0,3,192,0,3,192,0,3
1260 DATA192,0,3,192,0,3,192,0,3
```

```
1997 REM ****************************************
1998 REM *   SPRITE 0 VORBESETZEN              *
1999 REM ****************************************
2000 POKEV+21,5
2010 POKE2040,13
2020 FORI=1TO63
2030 READPU
2040 POKE831+I,PU
2050 NEXT
2052 X=30
2054 Y=50
2060 SPRITE,0,X,Y
2070 GRENZEN,0,20,50,320,230,0
2080 POKEV+39,7
2090 POKEV+28,1
2097 REM ****************************************
2098 REM *   SPRITE 1 VORBESETZEN              *
2099 REM ****************************************
2100 POKE2041,14
2110 FORI=1TO63
2120 READPU
2130 POKE895+I,PU
2135 NEXT
2140 X1=100
2150 Y1=100
2160 SPRITE,1,X1,Y1
2170 GRENZEN,1,20,50,320,230,0
2180 POKEV+40,5
2197 REM ****************************************
2198 REM *   SPRITE 2 VORBESETZEN              *
2199 REM ****************************************
2200 POKE2042,15
2210 FORI=1TO63
2220 READPU
2230 POKE959+I,PU
2235 NEXT
2240 X2=200*RND(-TI)+100
2250 Y2=150*RND(-TI)+100
2260 SPRITE,2,X2,Y2
2270 REM GRENZEN,2,10,20,330,190,15
2271 GRENZEN,2,10,20,330,100,15
2280 POKEV+41,1
2997 REM ****************************************
2998 REM *   BEWEGUNGSABLAUF                   *
2999 REM ****************************************
3000 PRINT"□"
3005 G=125
3006 FAHRT,2,150,45
3010 GETB$
3014 REM ZA=ZA+1:IFZA>5THENGOSUB6000
3016 IFPEEK(V+30)<>0THENGOSUB7000
```

```
3018 IF L=1 AND PEEK(V+31)<>0 THEN GOSUB 8530
3019 IF L=1 AND PEEK(V)>238 AND PEEK(V)<242 AND
                    PEEK(V+1)>180THEN GOSUB9030
3020 IFB$=""THEN3010
3030 IFB$="S"THEN:FAHRT,0,0,0:GOTO3010
3040 IFB$="D"THEN:FAHRT,0,G,0:GOTO3010
3050 IFB$="E"THEN:FAHRT,0,G,45:GOTO3010
3060 IFB$="W"THEN:FAHRT,0,G,90:GOTO3010
3070 IFB$="Q"THEN:FAHRT,0,G,135:GOTO3010
3080 IFB$="A"THEN:FAHRT,0,G,180:GOTO3010
3090 IFB$="Z"THEN:FAHRT,0,G,225:GOTO3010
3100 IFB$="X"THEN:FAHRT,0,G,270:GOTO3010
3110 IFB$="C"THEN:FAHRT,0,G,315:GOTO3010
3120 IFB$="↑"THENGOSUB5000:GOTO3010
3130 IFB$="N"THENPOKEV+21,5:GOTO3010
3135 IFB$="G"THENGOSUB3500:GOTO3010
3136 IFB$="L"THENGOSUB8030
3138 IFB$="V"ORB$="K"THENGOSUB9500
3140 GOTO3010
3500 GETA$:IFA$=""THEN3500
3510 A=VAL(A$)
3520 IFA<1THEN3500
3530 G=A*50
3540 RETURN
3997 REM ******************************************
3998 REM *   SPRITE 0 BEWEGEN                     *
3999 REM ******************************************
4000 IFSTHENGOSUB5100
4005 IFX<0THENX=0:GOTO4030
4010 IFX>255THENX=255:GOTO4030
4020 POKEV,X
4030 IFY<0THENY=0:RETURN
4040 IFY>200THENY=200:RETURN
4050 POKEV+1,Y
4060 RETURN
4997 REM ******************************************
4998 REM *   SPRITE 1 BEWEGEN                     *
4999 REM ******************************************
5000 POKEV+21,7
5010 S=PEEK(V)+256*(PEEK(V+16)AND1)
5020 SPRITE,1,S,PEEK(V+1)
5030 FAHRT,1,500,90
5040 RETURN
5200 S=0
5210 POKEV+21,5
5220 RETURN
5997 REM ******************************************
5998 REM *   SPRITE 2 BEWEGEN                     *
5999 REM ******************************************
6000 RETURN
6010 FAHRT,2,100+100*RND(2),360*RND(1)
```

```
6040 ZA=0
6099 RETURN
6997 REM ****************************************
6998 REM *   KOLLISIONSPRUEFUNG                 *
6999 REM ****************************************
7000 KO=PEEK(V+30)
7005 IFKO=3THENRETURN
7010 IFKO=6THEN7100
7020 IFKO=5THEN7200
7030 RETURN
7097 REM ****************************************
7098 REM *   SPRITE 1 GETROFFEN                 *
7099 REM ****************************************
7100 FORF=0TO15:FORWA=1TO15
7105 POKEV+21,5
7110 POKEV+41,F
7120 NEXT:NEXT
7130 POKEV+21,1
7140 S=0
7150 RETURN
7197 REM ****************************************
7198 REM *   SPRITE 2 GETROFFEN                 *
7199 REM ****************************************
7200 FORF=0TO15:FORWA=1TO15
7210 POKEV+41,F
7220 POKEV+39,15-F
7230 NEXT:NEXT
7235 POKEV+39,7
7237 X=X+50
7238 Y=Y+50
7240 RETURN
8000 REM ****************************************
8010 REM *   LANDUNG                            *
8020 REM ****************************************
8030 H$(1)="                  "
8040 H$(2)="                  "
8050 H$(3)="                  "
8060 H$(4)="              "
8070 H$(5)="              "
8080 H$(6)="              "
8090 H$(7)="              "
8100 POKE V+21,1
8110 REM ------ RAKETE GEGEN GRENZEN FAHREN --------
8120 GRENZEN,0,20,50,200,230,0
8130 FAHRT,0,G,45
8140 REM ------ BASIS IN BILD SCHIEBEN ------------
8150 FORI=1TO25
8160 PRINT" ",
8170 H1$=H1$+" "
8180 PRINT"                   ";
8190 FORJ=1TO7
```

```
8200 PRINTH1$;H$(J)
8210 NEXT
8220 REM ----- LAENGER WERDENDE WARTESCHLEIFE ----
8230 FORJ=1TO30*I
8240 NEXT
8250 NEXT
8260 REM ----- ANFLUGVORBEREITUNG ---------------
8270 FAHRT,0,0,0
8280 L=1
8290 GRENZEN,0,20,50,320,230,0
8300 RETURN
8500 REM ****************************************
8510 REM *   BEHANDLUNG MISSGLUECKTE LANDUNG     *
8520 REM ****************************************
8530 FAHRT,0,0,0
8540 REM ----- BLINKEN SPRITE / BASIS -----------
8550 FORI=1TO10
8560 POKEV+39,1
8570 PRINT"■";
8580 PRINT"SQQQQQQQQQQQQQQQQQQQ";
8590 FORJ=1TO7
8600 PRINTH1$;H$(J)
8610 NEXT
8620 POKEV+39,0
8630 PRINT"■";
8640 PRINT"SQQQQQQQQQQQQQQQQQQQ";
8650 FORJ=1TO7
8660 PRINTH1$;H$(J)
8670 NEXT
8680 NEXT
8690 REM ----- ABSCHLUSSBEHANDLUNG --------------
8700 L=0
8710 POKEV+31,0
8720 PRINT"]
8730 POKEV+21,0
8740 PRINT"]QQQQQQQQQQQ]         LANDUNG MISSGLUECKT
8750 FORI=1TO2000
8760 NEXT
8770 POKEV+21,5
8780 POKEV+39,7
8790 PRINT"]
8800 H1$=""
8810 RETURN
9000 REM ****************************************
9010 REM *   BEHANDLUNG ERFOLGREICHE LANDUNG     *
9020 REM ****************************************
9030 L=0
9040 FAHRT,0,0,0
9050 PRINT"SQQQQQQQQQQQ      ■LANDUNG ERFOLGREICH
9060 FORI=1TO2000
9070 NEXT
```

```
9080 POKEV+21,5
9090 H1$=""
9100 PRINT"⊐
9110 RETURN
9500 VK=0
9510 GETA$
9520 IFA$=""THEN9510
9530 IFA$="X"THENVK=V+29
9540 IFA$="Y"THENVK=V+23
9550 IFVK=0THEN9510
9560 GETA$
9570 IFA$=""THEN9560
9580 A=VAL(A$)
9590 IFA<1ORA>8THEN9560
9600 KV=2↑(A-1)
9610 IFB$="V"THENPOKEVK,PEEK(VK) OR KV
9620 IFB$="K"THENPOKEVK,PEEK(VK) AND NOT KV
9630 RETURN
```

Anhang 2 : BALKEN 3 - Beispiel aus Band 3

```
1000 REM ************************************************
1010 REM *            B A L K E N   3                   *
1020 REM ************************************************
1030 IFL=0THENL=1:LOAD"BASERW3.OBJ",8,1
1040 IFL=1THENL=2:LOAD"GRA3.OBJ",8,1
1050 IFL=2THENL=3:LOAD"PAUSE.OBJ",8,1
1060 IFL=3THENL=4:LOAD"GRALM.OBJ",8,1
1070 SYS51200
1080 POKE53280,11
1090 POKE53281,11
1100 V=53248
2000 REM ************************************************
2010 REM *     DATEN ERFASSEN UND VORBEREITEN           *
2020 REM ************************************************
2030 INPUT"{CLR}ANZAHL DER BALKEN";AB
2040 DIMWB(AB,3)
2050 DIMHO(AB,3)
2060 REM----- WERTE DER BALKEN ERFASSEN -----------------
2070 FORI=1TOAB
2080 FORJ=1TO3
2090 PRINTI;"TER BALKEN ";J;"-TER EINTRAG";
2100 INPUTWB(I,J)
2110 WB(I,0)=WB(I,0)+WB(I,J)
2120 NEXT
2130 NEXT
2140 REM----- MAXIMUM BESTIMMEN -------------------------
2150 FORI=1TOAB
2160 IFWB(I,0)>MATHENMA=WB(I,0)
2170 NEXT
2180 REM----- GRAFISCHE AUSGABEDATEN ERFASSEN -----------
2190 INPUT"ABSTAND                ";A
2200 INPUT"BREITE                 ";BR
2210 INPUT"GRUNDZEILE             ";GZ
2220 INPUT"MAXIMALE HOEHE (ZEILE) ";MH
2230 INPUT"ERHOEHUNG              ";EH
2240 INPUT"BEGINN IN SPALTE       ";BS
2250 REM----- BALKENHOEHEN NORMIEREN --------------------
2260 FORI=1TOAB
2270 HO(I,0)=INT(WB(I,0)*(GZ-MH-EH*(AB-1))/MA+.5)
2280 NEXT
2290 REM----- GROESSE DER TEILBALKEN BESTIMMEN ----------
2300 FORI=1TOAB
2310 FORJ=1TO3
2320 HO(I,J)=HO(I,0)*WB(I,J)/WB(I,0)
2330 NEXT
2340 NEXT
3000 REM ************************************************
3010 REM *     AUSGABE DER BALKEN                       *
3020 REM ************************************************
```

```
3030 GREIN,7,1,8
3040 UX=BS-A
3050 FORI=1TOAB
3060 OX=UX+A
3070 UX=OX+BR
3080 OY=GZ-HO(I,1)-EH*(I-1)
3090 UY=GZ-EH*(I-1)
3100 BLOCK,1,OX,OY,UX,UY
3110 OY=GZ-HO(I,1)-HO(I,2)-EH*(I-1)
3120 UY=GZ-HO(I,1)-EH*(I-1)
3130 BLOCK,2,OX,OY,UX,UY
3140 OY=GZ-HO(I,1)-HO(I,2)-HO(I,3)-EH*(I-1)
3150 UY=GZ-HO(I,1)-HO(I,2)-EH*(I-1)
3160 BLOCK,3,OX,OY,UX,UY
3170 NEXT
3180 GETA$:IFA$=""THEN3180
3190 GRAUS
3200 END
60000 SAVE"@:BALKEN",8
```

Anhang 3 : Übersicht Parametertypen:

A - Ausgabeparameter von diesem Unterprogramm
E - Eingabeparameter für dieses Unterprogramm
G - Globale Variable
H - Hilfsvariable
P - Aufrufparameter an Unterprogramm
R - Rückgabeparameter von Unterprogramm
T - Transienter Parameter (ist in einem Unterprogramm gleichzeitig Eingabeparameter (E) und Aufrufparameter (P) bzw. A und R

Globale Variablen gibt es zwar bei Basic nicht, da Basic keine block-orientierte Sprache wie z.B. PL/1 ist, aber in diesem Buch ist dieser Begriff für Variablen verwendet worden, die man in anderen Sprachen global definiert hätte.

Da der verwendete Typenraddrucker die Zeichen 'größer als' und 'kleiner als' nicht drucken kann, wurden diese Zeichen wie folgt ersetzt:

NE - Not Equal	/ ungleich
GT - Greater Than	/ größer als
LT - Less Than	/ kleiner als
GE - Greater or Equal	/ größer gleich
LE - Less or Equal	/ kleiner gleich

Die in den Kapiteln angegebenen Zeilenbereiche können mit denen der abgedruckten Listings differieren, weil offensichtliche REM-Zeilen (in der Regel die Überschriften von Unterprogrammen) nicht mitgedruckt wurden.

Anhang 4 : Übersicht der Bildschirmcodes

BILDSCHIRMSTEUERCODES

ZEICHEN	CODE
CURSOR NACH UNTEN	[illegible]
CURSOR NACH OBEN	[illegible]
CURSOR NACH RECHTS	[illegible]
CURSOR NACH LINKS	[illegible]
DEL (ZEI. LOESCHEN)	[illegible]
HOME (CRSR OBEN LINKS)	[illegible]
CLR (BILDSCHIRM LOESCHEN)	[illegible]
REVERSE EIN	[illegible]
REVERSE AUS	[illegible]

Anhang 5 : Befehlsübersicht mit Parameter

- : kein Parameter;
'*' vor den Parametern bedeutet Funktion

Programmierhilfen:

AUTO	Startnummer,Schrittweite
COLD	-
DELAY	Listgeschwindigkeit (1-255)
DISAPA	:
DISPLAY	-
DUMP	-
FIND	Basic-Code oder Zeichenreihe
KEY	Nummer,'Befehl'
MERGE	"Programmname",Gerätenummer
OLD	-
OPTION	10 (Ein) ; OPTION andere Zahl (Aus)
PAGE	Anzahl der Zeilen - 1
RENUMBER	erste Zeilennummer,Schrittweite
RETRACE	-
SECURE	0
TRACE	10 (Ein); Trace 0 (Aus)

Struktur-Befehle und ERROR-Befehle:

CALL	Marke (Label)
CGOTO	(Zeilennummer)
DISABLE	-
END PROC	-
ERRLN	* -
ERRN	* -
EXEC	Marke (Label)
GLOBAL	-
IF...THEN...: ELSE :	-
LOCAL	Variable 1,Variable 2,Variable 3,...
LOOP...EXIT IF ...END LOOP	-
NO ERROR	-
ON ERROR	-
ON KEY	Zeichenreihe,Anweisungen
OUT	-
PROC	Marke (Label)
RCOMP...ELSE	-
REPEAT...UNTIL	-
RESUME	-

Grafik-Befehle

Die verwendeten Abkürzungen haben folgende Bedeutung:

KMX - X-Koordinate Kreismittelpunkt
KMY - Y-Koordinate Kreismittelpunkt
RX - Radius in X-Richtung
RY - Radius in Y-Richtung
EWI - Endwinkel
SWI - Startwinkel
WI - Winkel
X1 - X-Koordinate oben/Anfang
X2 - X-Koordinate unten/Ende
Y1 - Y-Koordinate oben/Anfang
Y2 - Y-Koordinate unten/Ende
ZF - Zeichenfarbe

ANGL	KMX,KMY,WI,RX,RY,ZF
ARC	KMX,KMY,SWI,EWI,Abstand,RX,RY,ZF
BLOCK	X1,Y1,X2,Y2,ZF
CHAR	X1,Y1,Bildschirmcode,ZF,Größe
CIRCLE	KMX,KMY,RX,RY,ZF
CSET	n (0,1,2)
DRAW	Zeichenreihe,X1,Y1,ZF oder Variable,X1,Y1,ZF
GRAPHICS *	-
HICOL	-
HIRES	Punktfarbe,Hintergrundfarbe
LINE	X1,Y1,X2,Y2,ZF
LOW COL	Farbe 1,Farbe 2,Farbe 3
MULTI	Farbe 1,Farbe 2,Farbe 3
PAINT	X1,Y1,ZF
PLOT	X1,Y1,ZF
REC	X1,Y1,X2,X2sZF
ROT	Drehwinkel,Größe
TEST *	Variable = TEST(X1,Y1)
TEXT	X1,Y1,"Text",ZF,Größe,Abstand

Sprite-Befehle:

Die verwendeten Abkürzungen haben folgende Bedeutung:

G - Geschwindigkeit (1...255)
NR - MOB-Nummer, Sprite-Nummer
SX - Startkoordinate X
SY - Startkoordinate Y
ZX - Zielkoordinate X
ZY - Zielkoordinate Y

```
(Klammeraffe)   erlaubt sind: 'A','B','C','D','.',' '
CHECK      * (NR) oder (NR1,NR2)
CMOB         Farbe 1,Farbe 2
DESIGN       0 (norm.) oder 1 (MCM),Speicheranfangsadresse
DETECT       0 (MOB/MOB) oder 1 (MOB/Hintergrund)
MMOB         NR,SX,SY,ZX,ZY,Größe,G
MOB OFF      NR
MOB SET      NR,Block,Farbe,Priorität,Auflösung
RLOCMOB      NR,ZX,ZY,Größe,G
```

Musik-Befehle:

```
ENVELOPE     Stimme,Attack,Decay,Sustain,Release
MUSIC        Tondauer,"Noten"
PLAY         n (0,1,2)
SOUND      * -
VOL          n (0...15)
WAVE         Stimme,Wellenform
```

Befehle für Zeichenreihen

```
AT         * : PRINT AT(Spalte,Zeile)Zeichenreihe
CENTRE       Zeichenreihe
DUP        * (Zeichreihe,n)
INSERT     * ("Einzusetz. String","Äußer.String",p)
INST       * ("Überschr.String","Alter String",p)
PLACE      * ("gesuchter String",Zeichenreihe)
USE          Zeichereihe,Zeichenreihe
```

Befehl für Zahlen:

```
$          * Hexzahl (4-stellig)
%          * Binärzahl (8-stellig)
DIV        * (A,B)
EXOR       * (A,B)
FRAC       * (A,B)
MOD        * (A,B)
```

Bildschirmsteuerung

```
AT         * : PRINT AT(Spalte,Zeile)Zeichenreihe
BCKGNDS      Farbe 1,Farbe 2,Farbe 3,Farbe 4
BFLASH       Geschwindigkeit,Farbe 1,Farbe 2
BFLASH       0
COLOUR       Rahmenfarbe,Hintergrundfarbe
COPY         -
CSET         0, 1, 2
```

DOWNB	erste Z.,erste Sp.,letzte Z.,letzte Sp.
DOWNW	erste Z.,erste Sp.,letzte Z.,letzte Sp.
FCHR	Zeile,Spalte,Spaltenzahl,Zeilenzahl,Code
FCOL	Zeile,Spalte,Spaltenzahl,Zeilenzahl,Farbe
FLASH	Farbe,Geschwindigkeit
FILL	Zeile,Spalte,Spaltenzahl,Zeilenzahl,Code,Farbe
FLASH	Farbe,Geschwindigkeit
HRDCPY	-
INV	Zeile,Spalte,Spaltenzahl,Zeilenzahl
LEFTB	erste Z.,erste Sp.,letzte Z.,letzte Sp.
LEFTW	erste Z.,erste Sp.,letzte Z.,letzte Sp.
MEM	-
MOVE	Zeile,Spalte,Spaltenz.,Zeilenz.,ab Zei.,ab Spal.
NRM	-
OFF	-
RIGHTB	erste Z.,erste Sp.,letzte Z.,letzte Sp.
RIGHTW	erste Z.,erste Sp.,letzte Z.,letzte Sp.
SCRLD	2,8,2,"Name" oder 1,1,1,"Name"
SCRSV	2,8,2,"Name,s,w" oder 1,1,1,"Name"
UPB	erste Z.,erste Sp.,letzte Z.,letzte Sp.
UPW	erste Z.,erste Sp.,letzte Z.,letzte Sp.

Befehle für Light-Pen, Joy-Stick und Paddle

JOY	*	-
PENX	*	-
PENY	*	-
POT	*	(0),(1)

Sonstige Befehle

DESIGN		2 (oder 3),Speicheranfangsadresse
DIR		"$Zeichenreihe
DISK		"Anweisungen
FETCH		"Kontrolltaste",Anzahl Zeichen,Variable
INKEY	*	-
LIN	*	-
MEM		-
PAUSE		Zeit in Sekunden
RESET		Zeilennummer

Raum für Ihre Notizen

Anhang 6 : Befehlsübersicht mit Fundstellen

Befehl	Bedeutung	Seite Handbuch	Seite in diesem Buch
A			
ANGL	Radius zeichnen	33	23,226
ARC	Segment zeichnen	31	23,43,219
AT	Cursor auf Bildschirm positionieren		
AUTO	Zeilennummernvergabe bei Programmedit.		
B			
BCKGNDS	setzt vier Hintergrundfarben und schaltet auf Extended-Color-Mode	-	32,218
BFLASH	Farbwechsel Bildschirmrahmen einschalten	41	28,174
BFLASH 0	Farbwechsel Bildschirmrahmen ausschalten	41	28
BLOCK	farbig ausgefülltes Rechteck ausgeben	34	23,39,56,269
C			
CALL	Sprung zu einer mit PROC definierten Routine (ähnlich GOTO)	60	21,189
CENTRE	Ausgabe einer Zeichenreihe in deer Mitte	18	128
CGOTO	Unbedingeter Sprung einer Bildschirmzeile	10	76,183
CHAR	Zeichen in Grafik-Bildschirm	39	24,51,203
CHECK	Kollision abfragen	54	26,91,162

CIRCLE	Ellipse (Sonderfall: Kreis) ausgeben	30	23,171,217
CMOB	Farben für Multi-Color-Sprite festlegen	48	25,83,81,91,217
COLD	Kaltstart, ersetzt aus-/einschalten	14	16,128
COLOUR	Setzt Rahmen- und Hintergrundfarbe	-	32,70,172
COPY	Hardcopy einer hochauflösenden Grafik	46	29,37,263
CSET	Zeichensatz umschalten	37	24,70,264
D			
DELAY	Listgeschwindigkeit einstellen	13	16,195
DESIGN	Speicherzuteilung für Sprite oder	47,55	25,74,82,91,108
	neu zu erstellendes Zeichen festlegen		108,109,214
DETECT	Kollision vorbereiten	54	26,91,273
DIR	Inhaltsverzeichnis einer Diskette ganz	24	30,172
	oder teilweise (Jokerzeichen) anzeigen		
DISAPA	Anweisung schützen	14	16,163
DISK	Discbefehl ausführen	23	30,247
DISABLE	setzt ON KEY außer Kraft	-	32,193
DISPLAY	Belegung der Funktionstasten anzeigen	9	17,274
DIV	Division ohne Rest	21	28,157
DOWNB,DOWNW	Bildschirmbereich nach unten rollen	-	29,235,237
DRAW	Figur zeichnen	36	24,47,200
DUMP	Variablen mit Werten anzeigen	14	17,197
DUP	Zeichenreihe vervielfachen	18	107,153
E			
END PROC	Ende einer Routine,ähnlich RETURN	59	21,192
ENVELOPE	Hüllkurve einstellen	68	115,182
ERRLN	Zeile, wo ein Fehler auftritt	63	21,163
ERRN	Fehlernummer	63	21,163
EXEC	Unterprogrammaufruf für Routinen die mit		

Befehl	Bedeutung	Seite Handbuch	Seite in diesem Buch
	PROC und END PROC definiert wurden		
EXIT	Verlassen einer Schleife	58	192
EXOR	bitweise Verknüpfung von Zahlen mit EXKLUSIV ODER	23	27,113,153
F			
FCHR	Bildschirmbereich mit Zeichen füllen	41	28,206
FCOL	Zeichenfarbe im Bildschirmbereich bestimmen	41	28,206
FETCH	Kontrollierte Eingabe	20	30,238
FILL	Bildschirmbereich mit Farbe und Zeichen füllen	42	28,205
FIND	Basic-Befehle oder Zeichenreihen im Programm suchen	13	17,212
FLASH	Blinken einer Bildschirmfarbe ein-schalten	40	28,94,177,178
FRAC	Nachkommastellen einer Dezimalzahl	22	28,154
G			
GLOBAL	ursprünglichen Variablenwert wieder zuweisen	62	21,207
GRAPHICS	liefert Konstante $D000=53248 (VIC)	-	32,161
H			
HI COL	Nach LOW COL zum zurücksetzen auf die drei	28	23,69,205

HIRES	Farben, die mit MULTI definiert werden hochauflösende Grafik (mit Wahl der Vordergrund- und Hintergrundfarbe) einschalten	25	22,37,164
HRDCPY	Hardcopy eines normalen Bildschirmes	46	29,250
I			
IF..THEN..ELSE	Bedingte anweisung mit doppelter Anwendungsmöglichkeit	56	20,188
INKEY	Abfrage auf gedrückte Funktionstaste	20	31,161,179
INSERT	Zeichenreihe in andere einfügen	16	107,156
INST	Zeichenreihe mit einer anderen überschreiben	17	107,160
INV	Bildschirmbereich invertieren	43	28,231
J			
JOY	Abfrage Joystick	71	64,141
K			
KEY	Funktionstasten mit Basicbefehl belegen	8	17,252,180
L			
LEFTB,LEFTW	Bildschirm nach links rollen	-	29,234,235
LIN	Cursorposition feststellen	10	31,159
LINE	Linie zeichnen	29	23,44,59,167 207
LOCAL	Block bedingte Variablen	62	21,206
LOOP...EXIT IF	Schleifendurchlauf	58	21,108,195

Befehl	Bedeutung	Seite Hand-buch	Seite in diesem Buch
END LOOP	mit bedingtem Abbruch		
LOW COL	drei weitere Farben zum MULTI-COLOR-MODUS zuschalten	27	23,169
M			
MEM	Zeichensatz von ROM-Bereich in RAM-Bereich verlegen	-	31,273
MERGE	anderes Programm in bestehendes ein-kopieren	10	18,132
MMOB	Sprite darstellen oder bewegen	50	25,51,85,88,95 173
MOB OFF	Sprite ausschalten	50	25,88,95,219
MOB SET	Eigenschaften eines Sprite festlegen	49	25,51,84,92,176
MOD	'Modulo'	21	28,158
MOVE	Bildschirmbereich duplizieren	43	28,231
MULTI	MuLTI-COLOR-MODUS mit drei Zeichenfarben bestimmen	26	22,37,171
MUSIC	Noten festlegen	68	26,115,177
N			
NO ERROR	Fehlermeldungen unterdrücken	63	22,196
NRM	schaltet MEM und BCKGNDS ab	-	33,77,219
O			
OFF	Blinken einer Bildschirmfarbe ausschalten	41	95,219

OLD	NEW-Befehl aufheben	15	18,196
ON ERROR	Sprungverteile für Fehlermeldungen	63	21,195
ON KEY	Programmverzweigung nach Tastendruck	-	33,193
OPTION	Simon's Basic-Befehle hervorheben	12	18,187
OUT	Standardfehlermeldung ausgeben	-	196
P			
PAGE	seitenweise Listenausgabe	12	18,179
PAINT	Fläche mit Farbe füllen	34	24,49,66,253
PAUSE	Programmablauf anhalten	9	31,218
PENX	X-Koordinate des Light-Pen	71	142
PENY	Y-Koordinate des Light-Pen	71	142
PLACE	Zeichenreihe ersetzen	17	27,155,179
PLAY	Musikwiedergabe	69	26,115,146,181
PLOT	Punkt setzen	28	23,65,165,185
POT	Widerstand Paddle feststellen Potentionmeter	71	64,73
PROC	Sprungadresse (symbolisch)	59	21,107,108,113 197
R			
RCOMP	Bedingte Anweisung, wobei die Bedingung von der letzten IF-Abfrage übernommen wird	57	20,197
REC	Rechteck zeichnen	26	23,37,266
RENUMBER	Zeile umnumerieren (ohne Zeilenangabe bei GOTO und GOSUB)	9	19,133
REPEAT...UNTIL	ähnlich FOR-NEXT für bedingte Schleifen	57	21,108,113,186 197
RESET	Zeiger auf beliebige DATA-Zeile setzen	10	31,189

Befehl	Bedeutung	Seite Handbuch	Seite in diesem Buch
RESUME	beendet ON KEY-Anweisungsteil (RETURN)	-	33,194
RETRACE	Anzeigen des letzten TRACE-Fensters	14	187
RIGHTB,RIGHTW	Bildschirmbereich nach rechts rollen	44	29,93,237,138
RLOCMOB	Sprite bewegen	50	25,51,86,93,95 217
ROT	Figur drehen	37	24,47,142,202
S			
SCRLD	Bildschirm (der mit SCRSV gespeichert wurde)	46	29,243
SCRSV	Bildschirm (Normal-Modus) speichern	45	29,242
SECURE	Programmzeile schützen	15	19,195
SOUND	liefert Konstante $D400=54072 (SID)	-	33,115,161
T			
TEST	Punkt vorhanden	29	24,159,204
TEXT	Text in Grafik-Bildschirm	39	24,51,243
TRACE	aktuelle Zeilennummer, die im Programm durchlaufen wird, anzeigen	13	19,142,187
U			
UPB,UPW	Bildschirmbereich noch oben rollen	44	29,232,234
UNTIL	Schleifenende von REPEAT	57	108,113
USE	numerische Zeichenreihe formatieren	19	27,247

V			
VOL	Lautstärke einstellen	65	115,246
W			
WAVE	Wellenform einstellen	66	115,183
%	Binärzahl umwandeln	22	107,151,184
$	Hexzahl umwandeln	22	108,151

Anhang 7 : RE-SECURE

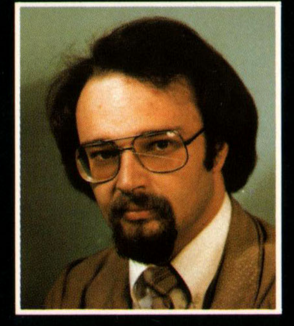
```
62000 A=2049
62010 LOOP
62020 VP=PEEK(A)+256*PEEK(A+1)
62030 EXIT IF VP=0
62040 IFPEEK(A+4)=0THENPOKEA+4,100:POKEA+5,65
62050 A=VP
62060 END LOOP
62070 END
```

Das Commodore 64-Buch

Band 5: Ein Leitfaden durch Simon's BASIC

Dieses Buch soll allen, die sich mit Simon's Basic beschäftigen, eine Hilfe sein. Sowohl für den Anfänger, der die Basic-Unterstützung nur zur Vereinfachung bei der eigenen Programmerstellung heranzieht, wie auch für den Profi, der Simon's Basic ändern möchte.
Entgegen dem Handbuch wurde eine andere Aufteilung gewählt. Zunächst werden alle Befehle kurz besprochen, wobei auf Besonderheiten, die nicht im Handbuch stehen, hingewiesen wird. Auch werden die Einsatzmöglichkeiten der Befehle eingehend beschrieben. In einem weiteren Teil werden die am häufigsten gebrauchten Befehle aus Simon's Basic anhand ausführlicher Beispiele näher erläutert (Grafik, Sprites, Musik), wobei die zahlreichen Abbildungen besonders die Grafikprogramme verdeutlichen sollen. Wußten Sie schon, daß 16 Farben gleichzeitig am Bildschirm dargestellt werden können? Den Abschluß bildet das kommentierte Assembler-Listing. Hingewiesen sei noch auf die Befehlsübersichten im Anhang, die Ihnen sicherlich gut als Nachschlagwerk dienen.

Auch die Bände 1 und 3 dieser Buchreihe beschäftigen sich zum Teil mit Basic-Erweiterungen, womit Sie gute Vergleichsmöglichkeiten über verschiedene Vorgehensweisen erhalten. Band 4 — Assembler / Disassembler — bietet die Grundlage für das kommentierte Assembler-Listing. In den Bänden 2 und 6 finden Sie etwas zur Entspannung: Spiele — Spiele — Spiele. Band 7 ist für Profis: Tips und Tricks wie zum Beispiel ein ausführliches Musikprogramm mit grafisch dargestellten Noten.





HANS LORENZ SCHNEIDER
geboren am 15.10.53 in Köln. Nach dem Abitur 1973 studierte er von 1976 bis 1980 Informatik an der Bundeswehrhochschule in München. Seit 1980 ist Schneider Inhaber und Geschäftsführer eines Software-Hauses, das sich hauptsächlich mit der Erstellung von Individual-Software für Mikrocomputer befaßt.

WERNER EBERL
geboren am 23.2.1962 in München, begann gleich nach dem Abitur 1980 sein Physik-Studium. Seine große Leidenschaft waren schon immer die Computer. Seit 1980 ist er auch als freier Mitarbeiter für ein Software-Haus tätig, wo er für die Umsetzung von Konzepten in lauffähige Programme verantwortlich ist.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-922120-71-7